Article

Not peer-reviewed version

Learning Optimal Strategies in a Duel Game

Angelos Gkekas , Athina Apostolidou , Artemis Vernadou , Athanasios Kehagias *

Posted Date: 11 November 2024

doi: 10.20944/preprints202411.0681.v1

Keywords: Duel game; tree game; backward induction; learning

Article

Learning Optimal Strategies in a Duel Game

An. Gkekas, Ath. Apostolidou, Art. Vernadou and Ath. Kehagias *

Aristotle University of Thessaloniki
* Correspondence: kehagiat@ece.auth.gr
‡ These authors contributed equally to this work.

Abstract: We study a duel game in which each player has partial knowledge of the game parameters. We present a method by which, in the course of repeated plays, each player estimates the missing parameters and consequently learns his optimal strategy.

Keywords: duel game; tree game; backward induction; learning

0. Introduction

We study a duel game in which certain game parameters (related to the players' *kill probabilities*) are unknown and present a method by which each player can estimate the opponent's parameters and consequently learn his optimal strategy.

The duel game with which we are concerned has been presented in [1] and a similar one in [2]. The game is a variation of duels and, more generally, *games of timing* studied in the literature [3–5].

The paper is organized as follows. In Section 1 we present the rules of the game. In Section 2 we solve the game under the assumption of complete information. In Section 3 we present an algorithm for solving the game when the players have incomplete information. In Section 4 we evaluate the algorithm by numerical experiments. Finally, in Section 5 we summarize our results and present our conclusions.

1. Game Description

The duel with which we are concerned is played between players P_1 and P_2, under the following rules.

1. It is played in discrete rounds (time steps) $t \in \{1, 2, ...\}$.
2. In the first turn, the players are at distance D.
3. P_1 (resp. P_2) plays on odd (resp. even) rounds.
4. On his turn, each player has two choices: (i) he can shoot his opponent or (ii) he can move one step forward, reducing their distance by one.
5. If P_n shoots, he has a *kill probability* $p_n(d)$ of hitting (and killing) his opponent, where d is their current distance. If he misses, the opponent can walk right next to him and shoot him for a certain kill.
6. Each player's payoff is 1 if he kills the opponent and -1 if he shoots and misses (in which case he is certain to be killed).

For $n \in \{1, 2\}$, we will denote by $x_n(t)$ the position of P_n at round t. The starting positions are $x_1(0) = 0$ and $x_2(0) = D$, with $D = 2N, N \in \mathbb{N}$. The distance between the players at time t is

$$d = |x_1(t) - x_2(t)|.$$

For $n \in \{1, 2\}$, the kill probability is a *decreasing* function $p_n : \{1, 2, \ldots D\} \to [0, 1]$ with $p_n(1) = 1$. It is convenient to describe the kill probabilities as vectors:

$$\mathbf{p}_n = (p_{n,1}, ..., p_{n,D}) = (p_n(1), ..., p_n(D)).$$

This duel can be modeled as an *extensive form game* or *tree game*. The game tree is a directed graph $G = (V, E)$ with vertex set

$$V = \{1, 2, ..., 2D\},$$

where

1. the vertex $v = d \in \{1, 2, ..., D\}$ corresponds to a game state in which the players are at distance d and
2. the vertex $v = d + D \in \{D + 1, D + 2, ..., 2D\}$ is a *terminal* vertex, in which the "active" player has fired at his opponent.

The edges correspond to *state transitions*; it is easy to see that the edge set is

$$E = \{(1, D + 1), (2, 1), (2, D + 2), (3, 2), (3, D + 3), ..., (D, D - 1), (D, 2D)\}.$$

An example of the game tree, for $D = 6$, appears in Figure 1. The circular (resp. square) vertices are the ones in which P_1 (resp. P_2) is active and the rhombic vertices are the terminal ones.

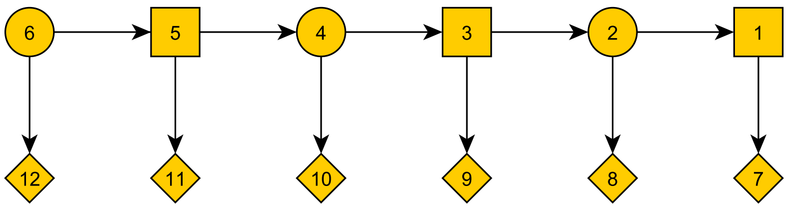

Figure 1. Game tree example

To complete the description of the game, we will define the *expected* payoff for the terminal vertices. Note that the terminal vertex $d + D$ is the child of the nonterminal vertex d in which:

1. The distance of the players is d and, assuming P_n to be the active player, his probability of hitting his opponent is $p_{n,d}$.
2. The active player is P_1 (resp. P_2) iff d is even (resp. odd).

Keeping the above in mind, we see that the payoff (to P_1) of vertex $d + D$ is

$$\forall d \in \{1, ..., D\} : Q(d + D) = \begin{cases} p_{1,d} \cdot 1 + (1 - p_{1,d}) \cdot (-1) = 2p_{1,d} - 1, & \text{when } d \text{ is even;} \\ p_{2,d} \cdot (-1) + (1 - p_{2,d}) \cdot 1 = 1 - 2p_{2,d} & \text{when } d \text{ is odd.} \end{cases}$$

The payoff to P_2 at vertex $d + D$ is $-Q(d + D)$. This completes the description of the duel.

2. Solution with Complete Information

It is easy to solve the above duel when each player knows D and *both* $\mathbf{p_1}$ and $\mathbf{p_2}$. We construct the game tree as described in Section 1 and we solve it by backward induction. Since the method is standard, we simply give an example of its application. Suppose that

$$\forall n, d : p_{n,d} = \begin{cases} 1 & \text{when } d = 1, \\ \min\left(1, \frac{c_n}{d^{k_n}}\right) & \text{when } d > 1. \end{cases}$$

We take $c_1 = 1, k_1 = 1, c_2 = 1, k_2 = \frac{1}{2}$. The kill probabilities are

Table 1. Kill probabilities

d	1	2	3	4	5	6
$p_{1,d}$	1.000	0.500	0.333	0.250	0.200	0.166
$p_{2,d}$	1.000	0.707	0.577	0.500	0.447	0.408

The game tree with terminal payoffs is illustrated in Figure 2.

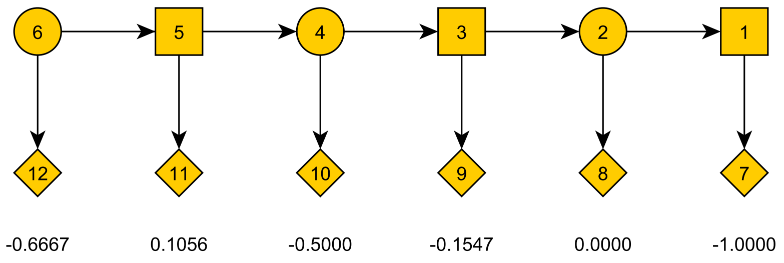

Figure 2. Game tree example with values of terminal vertices.

By the standard backward induction procedure we select the optimal action at each vertex and also compute the values of the nonterminal vertices. These are indicated in Figure 3 (optimal actions correspond to thick edges). We see that the game value is -0.1547, attained by P_2 shooting when the players are at distance 3 (which happens in round 4).

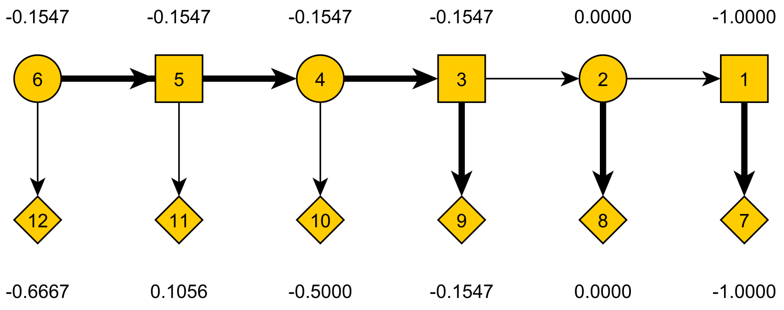

Figure 3. Game tree example with values of all vertices

We next present a proposition which characterizes each player's optimal strategy in terms of a *shooting threshold*. [1] In the following we use the standard notation by which "$-n$" denotes the "other player". I.e., $p_{-1} = p_2$ and $p_{-2} = p_1$.[2]

Theorem 1. We define for $n \in \{1,2\}$ the *shooting criterion* vectors $\mathbf{K}_n = (K_{n,1},...,K_{n,D})$ where

$$K_{n,1} = 1 \text{ and for } d \geq 2 : K_{n,d} = p_{n,d} + p_{-n,d-1}.$$

1 This proposition is stated informally in [1].
2 We use the same notation for several other quantities as will be seen in the sequel.

Then the optimal strategy for P_n is to shoot *as soon as it is his turn* and the distance of the players is less than or equal to the *shooting threshold* d_n, where:

$$d_1 = \max\{d : K_{1,d} \geq 1\}, \quad d_2 = \max\{d : K_{2,d} \geq 1\}.$$

Proof. Suppose the players are at distance d and the active player is P_n.

1. If P_{-n} will not shoot in the next round, when their distance will be $d-1$, then P_n must also not shoot in the current round, because he will have a higher kill probability in his next turn, when they will be at distance $d-2$.
2. If P_{-n} will shoot in the next round, when their distance will be $d-1$, then P_n should shoot iff $p_{n,d}$ (his kill probability now) is higher then $1-p_{-n,d-1}$ (P_{-n}'s miss probability in the next round). In other words, P_n must shoot iff
$$p_{n,d} \geq 1 - p_{-n,d-1}$$
or, equivalently, iff
$$K_{n,d} = p_{n,d} + p_{-n,d-1} \geq 1. \qquad (1)$$

Hence we can reason as follows.

1. At vertex 1, P_2 is active and his only choice is to shoot.
2. At vertex 2, P_1 is active and he knows P_2 will certainly shoot in the next round. Hence P_1 will shoot iff he has an advantage, i.e., iff
$$Q_1(2) \geq Q_1(1) \Leftrightarrow 2p_{1,2} - 1 \geq 1 - 2p_{2,1} \Leftrightarrow p_{1,2} + p_{2,1} \geq 1.$$
This is equivalent to $K_{1,1} \geq 1$ and will always be true.
3. Hence at vertex 3, P_2 is active and he knows that P_1 will certainly shoot in the next round (at vertex d). So P_2 will shoot iff
$$Q_1(3) \leq Q_1(2) \Leftrightarrow 1 - 2p_{2,3} \leq 2p_{1,2} - 1 \Leftrightarrow p_{2,3} + p_{1,2} \geq 1.$$
which is equivalent to $K_{2,3} \geq 1$. Also, if $K_{2,3} < 1$, then P_1 will know, when the game is at vertex 4, that P_2 will not shoot when at 3. So, P_1 will not shoot when at 4. But then, when at 5, P_2 knows that P_1 will not shoot when at 4. Continuing in this manner we see that $K_{2,3} < 1$ implies that firing will take place exactly at the vertex 2.
4. On the other hand, if $K_{2,3} \geq 1$, then P_1 knows when at 4 that P_2 will shoot at the next round. So, when at 4, P_1 should shoot iff $K_{1,4} \geq 1$. If, on the other hand, $K_{1,4} < 1$, then P_1 will not shoot when at 4 and P_2 will shoot when at 3.
5. We continue in this manner for increasing values of d. Since both $K_{1,d}$ and $K_{2,d}$ are decreasing with d, there will exist a maximum d_n value (it could equal D) in which some $K_{n,d}$ will be greater than one and P_n will be active; then P_n must shoot as soon as the game reaches or passes vertex d_n and he "has the action".

This completes the proof. □

Returning to our previous example, we compute the vectors K_n for $n \in \{1,2\}$ and list them in the following table.

Table 2. Shooting Criterion

d	1	2	3	4	5	6
Round	6	5	4	3	2	1
$K_{1,d}$		1.500	1.040	0.827	0.700	0.614
$K_{2,d}$		1.707	1.077	0.833	0.697	0.608

For P_1 the shooting criterion is last satisfied when the distance is $d = 3$; this happens at round 4 in which P_1 is inactive, so he should shoot at round 5. However, for P_2 the shooting criterion is also last satisfied at distance $d = 3$ and round 4, in which P_2 is active; so he should shoot at round 4. This is the same result we obtained with backward induction.

3. Solution with Incomplete Information

We will now present an approach to the solution of the duel when each player has incomplete knowledge of his opponent's kill probability function. More specifically, we assume that the players's kill probabilities have, for all d, the form $f(d;\theta)$ where θ is a parameter vector. Assuming P_1 has $\theta = \theta_1$ and P_2 has $\theta = \theta_2$, we have

$$\forall d : p_{1,d} = f(d;\theta_1), \quad p_{2,d} = f(d;\theta_2).$$

Furthermore, we assume that both players know the general form $f(d;\theta)$ and each P_n knows his own parameter vector θ_n but not the θ_{-n} of his opponent.

Obviously in this case the players cannot perform the computations of either backward induction or the shooting criterion. Consequently we will propose an "*exploration-and-exploitation*" approach. In other words, under the assumption that multiple duels will be played, each player initially adopts a random strategy and, collecting information from played games, he gradually builds and refines an estimate of his optimal strategy.

We give now a more detailed description of our approach, by Algorithm 1, presented below in pseudocode.

Algorithm 1 Learning the Optimal Duel Strategy

```
 1: Input: Duel parameters $D, \theta_1, \theta_2$; Learning parameters $\lambda, \sigma_0$; Number of plays $R$
 2: $\mathbf{p}_1$=CompKillProb($\theta_1$)
 3: $\mathbf{p}_2$=CompKillProb($\theta_2$)
 4: Randomly initialize parameter estimates $\theta_1^0, \theta_2^0$
 5: for $r \in \{1, 2, ..., R\}$ do
 6:     $\mathbf{p}_1^r$=CompKillProb($\theta_1^{r-1}$)
 7:     $\mathbf{p}_2^r$=CompKillProb($\theta_2^{r-1}$)
 8:     $d_1^r$=CompShootDist($\mathbf{p}_1, \mathbf{p}_2^r$)
 9:     $d_2^r$=CompShootDist($\mathbf{p}_1^r, \mathbf{p}_2$)
10:     $\sigma_r = \sigma_{r-1}/\lambda$
11:     X=PlayDuel($\mathbf{p}_1, \mathbf{p}_2, d_1^r, d_2^r, \sigma_r, X$)
12:     $(\widehat{\mathbf{p}}_1^r, \widehat{\mathbf{p}}_2^r)$=EstKillProb(X)
13:     $\theta_1^r$=EstPars($\widehat{\mathbf{p}}_1^r$, 1)
14:     $\theta_2^r$=EstPars($\widehat{\mathbf{p}}_2^r$, 2)
15: end for
16: return $d_1^R, d_2^R, \theta_1^R, \theta_2^R$
```

The following remarks explain the operation of the algorithm.

1. In line 1 the algorithm takes as input: (i) the duel parameters D, θ_1, θ_2, (ii) two *learning parameters* λ, σ_0 and (iii) the number R of duels used in the learning process.
2. Then, in lines 2-3, the true kill probability $\mathbf{p}_n$ (for $n \in \{1, 2\}$) is computed by the function CompKillProb(θ_n), which simply computes

$$\forall n, d : p_n(d) = f_n(d;\theta_n).$$

   We emphasize that these are the *true* kill probabilities.

3. In line 4 initial, randomly selected parameter vector estimates θ_1^0, θ_2^0 are generated.
4. Then the algorithm enters the loop of lines 5-15 (executed for R iterations) which constitutes the main learning process.

   (a) In lines 6-7 we compute new *estimates* of the kill probabilities $\mathbf{p}_n^r$, by function COMPKILLPROB, based on the estimates of parameters θ_n^{r-1}:

   $$\forall n, d : p_n^r(d) = f_n\left(d; \theta_n^{r-1}\right).$$

   We emphasize that these are *estimates* of the kill probabilities, based on the parameter estimates θ_n^{r-1}.

   (b) In lines 8-9 we compute new *estimates* of the shooting thresholds d_n^r, by function COMPSHOOTDIST. For P_n, this is achieved by computing the shooting criterion $\mathbf{K}_n$ using the (known to P_n) $\mathbf{p}_n$ and the (estimated by P_n) $\mathbf{p}_{-n}^r$.

   (c) In line 10, σ_r (which will be used as a standard deviation parameter) is divided by the factor $\lambda > 1$.

   (d) In line 11 the result of the duel is computed by the function PLAYDUEL. This is achieved as follows:

      i. For $n \in \{1, 2\}$, P_n selects a random shooting distance $\widehat{d}_n$ from the *discrete normal distribution* [6] with mean d_n^r and standard deviation σ_r.

      ii. With both $\widehat{d}_1, \widehat{d}_2$ selected, it is clear which player will shoot first; the outcome of the shot (hit or miss) is a Bernoulli random variable with success probability $p_{n,\widehat{d}_n}$, where P_n is the shooting player. Note that $\mathbf{p}_n$ is the *true* kill probability.

   The result is stored in a table X, which contains the data (shooting distance, shooting player, hit or miss) of every duel played up to the r-th iteration.

   (e) In line 12, the entire game records X are used by ESTKILLPROB(X) to obtain empirical estimates of the kill probabilities $\widehat{\mathbf{p}}_1^r, \widehat{\mathbf{p}}_2^r$. These estimates are as follows:

   $$\forall n, \forall d \in \mathcal{D}_n \colon \widehat{p}_{n,d}^r = \frac{\sum_{r \in \mathcal{R}_{n,d}} Z_r}{\left|\mathcal{R}_{n,d}\right|}$$

   where

   $$\begin{aligned} \mathcal{D}_n &= \{d : d \text{ such that } P_n \text{ may shoot}\} \\ \mathcal{R}_{n,d} &= \{r : \text{ in the } r\text{-th game } P_n \text{ actually shot from distance } d\}, \\ Z_r &= \begin{cases} 1 & \text{iff the shot in the } r\text{-th game hit the target,} \\ 0 & \text{iff the shot in the } r\text{-th game missed the target} \end{cases} \end{aligned}$$

   (f) In line 13-14 the function ESTPARS uses a least squares algorithm to find (only for the P_n who currently has the action) θ_n^r values which minimize the squared error

   $$J(\theta_n) = \sum_{d \in \mathcal{D}_n} \left(f_n(d; \theta_n) - \widehat{p}_{n,d}^r\right)^2.$$

   (g) In line 21 the algorithm returns the final estimates of optimal shooting distances d_1^R, d_2^R and parameters θ_1^R, θ_2^R.

Note that multiplication by $1/\lambda \in (0, 1)$ results in $\lim_{r \to \infty} \sigma_r = 0$. Hence, while in the initial iterations of the learning process the players essentially use random shooting thresholds (exploration) with standard deviation σ_r, it is hoped that, as r increases and σ_r of the used shooting thresholds goes to zero, the estimates of the kill probabilities and shooting thresholds will converge to their optimal values (exploitation). This is actually corroborated by the experiments we present in the next section.

4. Experiments

In this section we present numerical experiments to evaluate our approach.

4.1. Experiments Setup

In the following subsection we present several experiment groups, all of which share the same structure. Each experiment group corresponds to a particular form of the kill probability functions. In each case, the kill probability parameters, along with the initial player distance D, are the *game parameters*. For each choice of game parameters we proceed as follows.

First we select the learning parameters λ, σ and the number of learning steps R. These, together with the game parameters, are the *experiment parameters*. Then we select a number J of estimation experiments to run for each choice of experiment parameters. For each of the J experiments we compute the following quantities.

1. The relative error of the final kill probability parameter estimates. For a given parameter $\theta_{n,i}$, this error is defined to be
$$\Delta\theta_{n,i} = \left|\frac{\theta_{n,i} - \theta^R_{n,i}}{\theta_{n,i}}\right|.$$
2. The relative error of the shooting threshold estimates. Letting d^R_n be the estimate of the shooting threshold based on the true kill probability vector p_n and kill probability vector estimate p^R_{-n}, this error is defined to be
$$\Delta d_n = \left|\frac{d_n - d^R_n}{d_n}\right|.$$
3. The relative error of the optimal payoff estimates. Letting Q^R_n be the estimate of the optimal payoff (computed from the estimated shooting thresholds d^R_1, d^R_2) , this error is defined to be
$$\Delta Q_n = \begin{cases} \left|\frac{Q_n - Q^R_n}{Q_n}\right| & \text{iff } Q_n \neq 0 \\ 0 & \text{iff } Q_n = 0 \text{ and } Q^R_n = Q_n \\ 1 & \text{iff } Q_n = 0 \text{ and } Q^R_n \neq Q_n \end{cases}$$
Note that $\Delta Q_2 = \Delta Q_1$, because the game is zero-sum.

4.2. Experiment Group A

In this group the kill probability function has the form:

$$n \in \{1,2\} : p_{n,d} = \min\left(\frac{c_n}{d^{k_n}}, 1\right).$$

Let us look at the final results of a representative run of the learning algorithm. With $c_1 = 1$, $k_1 = 0.5$, $c_2 = 1$,$k_2 = 1$ and $D = 10$, we run the learning algorithm with $R = 1500$, $\sigma_0 = 6D$ and for three different values $\lambda \in \{1.001, 1.01, 1.05\}$. In Figure 4 we plot the logarithm (with base 10) of the relative payoff error $\Delta Q_1 + \epsilon$ (we have added $\epsilon = 10^{-3}$ to deal with the logarithm of zero error). The three curves plotted correspond to the λ values 1.001, 1.01, 1.05. We see that, for all λ values, the algorithm achieves zero relative error; in other words it learns the optimal strategy for both players. Furthermore convergence is achieved by the 1500-th iteration of the algorithm (1500-th duel played), as seen by the achieved logarithm value -3 (recall that we have added $\epsilon = 10^{-3}$ to the error, hence the true error is zero). Convergence is fastest for the largest λ value, i.e., $\lambda = 1.05$, and slowest for the smallest value $\lambda = 1.001$.

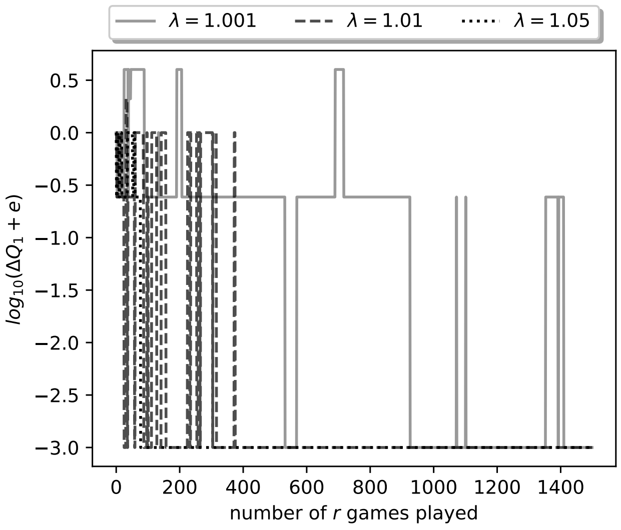

Figure 4. Plot of logarithmic relative error $\log_{10} \Delta Q_1$ of P_1's payoff for a representative run of the learning process.

In Figure 5 we plot the logarithmic relative errors $\log_{10} \Delta d_n$. These also, as expected, have converged to zero by the 1500-th iteration.

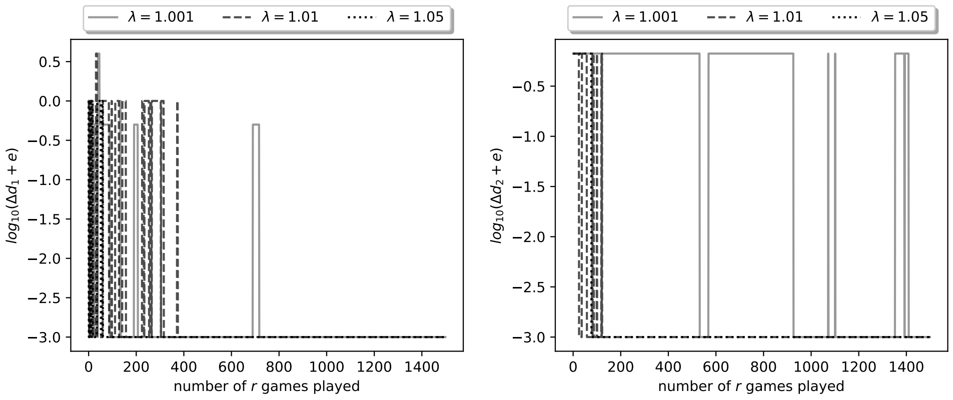

Figure 5. Plot of logarithmic relative errors $\log_{10} \Delta d_1$, $\log_{10} \Delta d_2$ for a representative run of the learning process.

The fact that the estimates of the optimal shooting thresholds and strategies achieve zero error, does not imply that the same is true of the kill probability parameter estimates. In Figure 6 we plot the relative errors Δc_1, Δk_1.

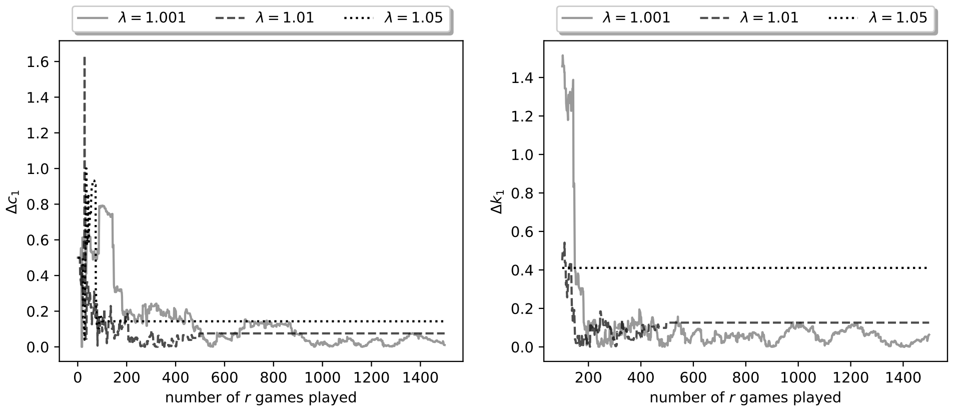

Figure 6. Plot of relative parameter errors Δc_1, Δk_1 for a representative run of the learning process.

It can be seen that these errors do not converge to zero; in fact, for $\lambda \in \{1.01, 1.05\}$ the errors converge to fixed nonzero values, which indicates that the algorithm obtains wrong estimates. However, the error is sufficiently small to still result in zero-error estimates of the shooting thresholds. The picture is similar for the errors Δc_2, Δk_2, hence their plots are omitted.

In the above we have given results for a particular run of the learning algorithm. This was a successful run, in the sense that it obtained zero-error estimates of the optimal strategies (and shooting thresholds). However, since our algorithm is stochastic, it is not guaranteed that every run will result in zero-error estimates. To better evaluate the algorithm, we have run it for $J = 10$ times and averaged the obtained results. In particular, in Figure 7 we plot the average of ten curves of the type plotted in Figure 4. Note that now we plot the curve for $R = 5000$ plays of the duel.

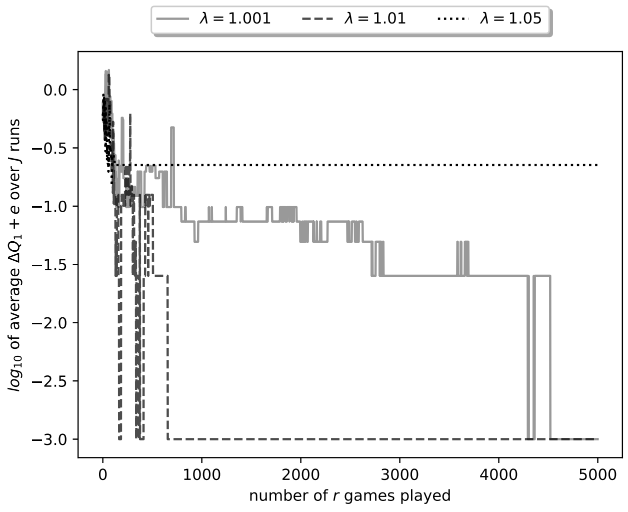

Figure 7. Plot of Q_1 (P_1's payoff) for a representative run of the learning process.

Several observations can be made regarding Figure 7.

1. For the smallest λ value, namely $\lambda = 1.001$, the respective curve reaches -3 at $r = 4521$. This corresponds to zero *average* error, which means that, in some algorithm runs, it took more than 4500 iterations (duel plays) to reach zero error.
2. For the $\lambda = 1.01$, all runs of the algorithm reached zero-error after $r = 656$ runs.
3. Finally, for $\lambda = 1.05$, the average error never reached zero; in fact 3 out of 10 runs converged to nonzero-error estimates, i.e., to non-optimal strategies.

The above observations corroborate a fact well known in the study of reinforcement learning [7]. Namely, a small learning rate (in our case small λ) results in higher probability of converging to the true parameter values, but also in slower convergence. This can be explained as follows: a small λ results in higher σ_r values for a large proportion of duels played by the algorithm; i.e., in more extensive *exploration*, which however results in slower *exploitation* (convergence). [3]

We conclude this group of experiments by running the learning algorithm for various combinations of game parameters; for each combination we record the average error attained at the end of the algorithm (i.e., at $r = R = 5000$). The results are summarized at the following tables.

[3] An analogous phenomenon occurs in connection to the "temperature parameter" T in simulated annealing [8] and many other learning algorithms.

Table 3. Values of final average relative error ΔQ_1 for $c_2 = 1, k_2 = 1$ and various values of c_1, k_1, λ. D is fixed at $D = 10$.

λ	1.001			1.01			1.05		
k_1 / c_1	0.50	1.00	1.50	0.50	1.00	1.50	0.50	1.00	1.50
1.00	0.000	0.000	0.000	0.000	0.100	0.482	0.224	0.500	1.207
1.50	0.000	0.000	0.000	0.059	0.049	1.748	0.215	0.480	3.777
2.00	0.000	0.000	0.048	0.074	0.199	0.097	0.144	0.980	0.072

Table 4. Round at which ΔQ_1 converged to zero for all $J = 10$ sessions for $c_2 = 1, k_2 = 1$ and various values of c_1, k_1, λ. D is fixed at $D = 10$. If ΔQ_1 did not converge for all sessions we note for how many sessions it converged.

λ	1.001			1.01			1.05		
k_1 / c_1	0.50	1.00	1.50	0.50	1.00	1.50	0.50	1.00	1.50
1.00	4521	1456	2983	656	9/10	8/10	7/10	5/10	5/10
1.50	3238	2939	1754	7/10	9/10	9/10	0/10	8/10	6/10
2.00	4153	423	8/10	2/10	9/10	6/10	3/10	4/10	6/10

From tables 6 and 7 we see that for $\lambda = 1.001$ almost all learning sessions conclude in zero ΔQ_1, while increasing the value of λ results in more sessions concluding with non-zero error estimates. Furthermore, we observe that when the average ΔQ_1 converges to zero for multiple values of λ the convergence is faster for bigger λ. These results highlight the trade off between *exploration* and *exploitation* discussed above.

Finally, in table 8 we see how many learning sessions were run and how many converged to zero error estimate ΔQ_1 for different values of D and λ.

Table 5. Fraction of learning sessions that converged to $\Delta Q_1 - 0$ for different values of D and λ.

λ / D	1.001	1.01	1.05
8	163/170	144/170	89/170
10	165/170	139/170	81/170
12	161/170	123/170	68/170
14	164/170	134/170	73/170
total	653/680	540/680	311/680

Table 6. Values of final average relative error ΔQ_1 for $d_{21} = 1, d_{22} = D$ and various values of d_{11}, d_{12}, λ. D is fixed at $D = 10$.

d_{11}	λ / d_{12}	1.001	1.01	1.05
1	$D/2$	0.000	0.233	1.000
1	$2D/3$	0.000	0.000	0.799
1	D	$\sim 6{\cdot}10^{-16}$	$\sim 2{\cdot}10^{-16}$	0.800
$D/3$	$2D/3$	0.000	0.000	2.799
$D/3$	D	0.000	0.000	0.777
$D/2$	D	0.000	0.000	0.480

Table 7. Round at which ΔQ_1 converged to zero for all $J = 10$ sessions for $d_{21} = 1$, $d_{22} = D$ and various values of d_{11}, d_{12}, λ. D is fixed at $D = 10$. If ΔQ_1 did not converge for all sessions we note for how many sessions it converged.

	λ	1.001	1.01	1.05
d_{11}	d_{12}			
1	$D/2$	287	7/10	0/10
1	$2D/3$	812	333	9/10
1	D	7/10	9/10	9/10
$D/3$	$2D/3$	326	139	5/10
$D/3$	D	225	397	5/10
$D/2$	D	711	464	6/10

Table 8. Fraction of learning sessions that converged to $\Delta Q_1 - 0$ for different values of D and λ.

λ	1.001	1.01	1.05
D			
8	76/110	80/110	49/110
10	97/110	95/110	56/110
12	94/110	78/110	39/110
14	100/110	84/110	40/110
total	367/440	337/440	184/440

4.3. Experiment Group B

In this group the kill probability function is piecewise linear

$$p_{n,d} = \begin{cases} 1 & \text{when } d \in [1, d_{n1}] \\ -\frac{1}{d_{n2}-d_{n1}}d + \frac{d_{n2}}{d_{n2}-d_{n1}} & \text{when } d \in [d_{n1}, d_{n2}] \\ 0 & \text{when } d \in [d_{n2}, D] \end{cases}$$

Let us look again at the final results of a representative run of the learning algorithm. With $d_{11} = D$, $d_{12} = D/3$, $d_{21} = 1$, $d_{22} = D$ and $D = 8$, we run the learning algorithm with $R = 500$, $\sigma_0 = 6D$ and for the values $\lambda \in \{1.001, 1.01, 1.05\}$. In Figure 8 we plot the logarithm (with base 10) of the relative payoff error $\Delta Q_1 + \epsilon$. We see similar results as in Group A, for all λ values, the algorithm achieves zero relative error. Convergence is achieved by the 300-th iteration of the algorithm and it is fastest for $\lambda = 1.05$, and slowest for $\lambda = 1.001$.

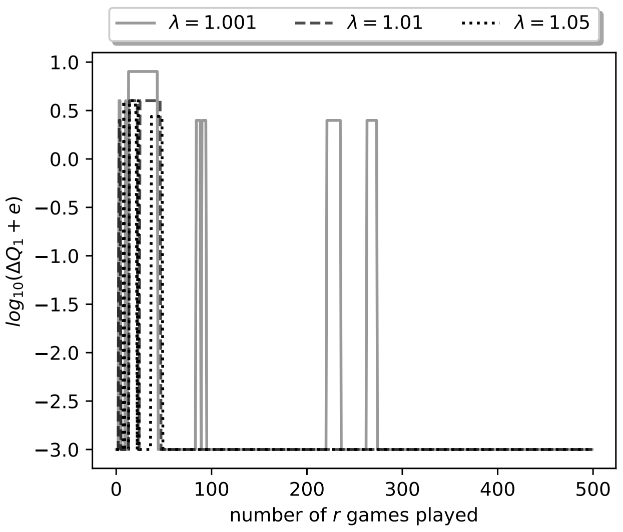

Figure 8. Plot of logarithmic relative error $\log_{10} \Delta Q_1$ of P_1's payoff for a representative run of the learning process.

In Figure 9 we plot the logarithmic relative errors $\log_{10} \Delta d_n$. These also, as expected, have converged to zero by the 300-th iteration.

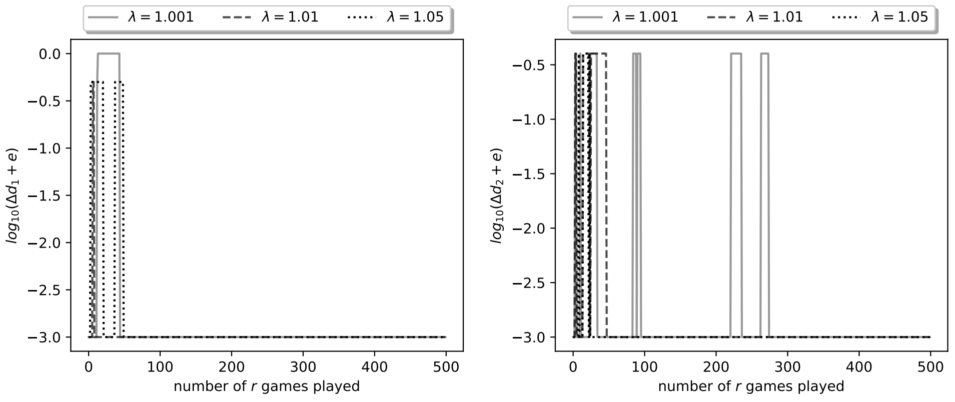

Figure 9. Plot of logarithmic relative errors $\log_{10} \Delta d_1$, $\log_{10} \Delta d_2$ for a representative run of the learning process.

As in Group A, the fact that the estimates of the optimal shooting thresholds and strategies achieve zero error does not imply that the same is true for the kill probability parameter estimates. In Figure 10, we plot the relative errors Δd_1 and Δd_2. The relative error Δd_2 converges to zero quickly, but the relative error Δd_1 does not converge to zero for $\lambda = 1.05$. For $\lambda = 1.05$ the error converges to a fixed nonzero value, which indicates that the algorithm obtains wrong estimates. However, the error is sufficiently small to still result in zero-error estimates of the shooting thresholds.

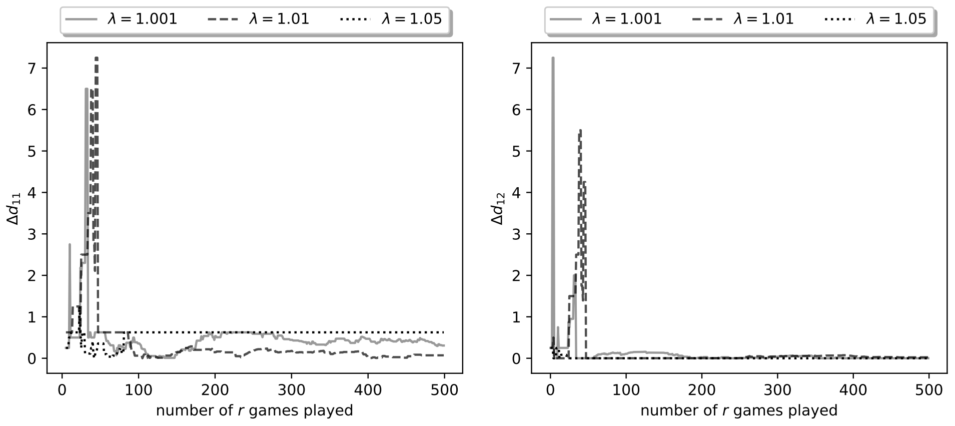

Figure 10. Plot of relative parameter errors Δd_1, Δd_2 for a representative run of the learning process.

As in group A, to better evaluate the algorithm, we have run it for $J = 10$ times and averaged the obtained results. In particular, in Figure 11 we plot the average of ten curves of the type plotted in Figure 4. Note that now we plot the curve for $R = 500$ plays of the duel.

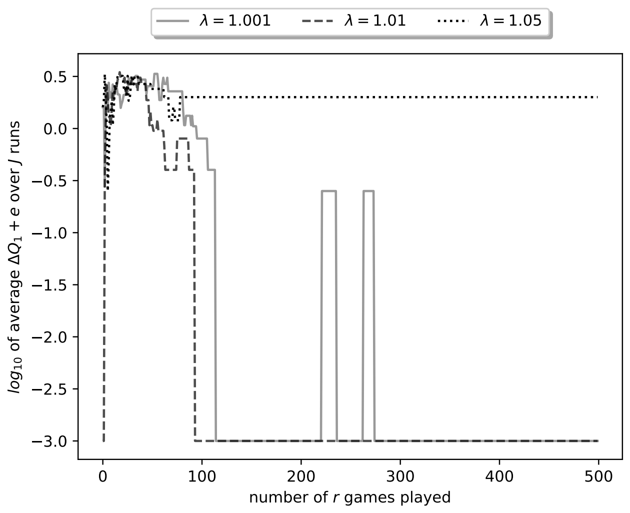

Figure 11. Plot of Q_1 (P_1's payoff) for a representative run of the learning process.

We again run the learning algorithm for various combinations of game parameters and record the average error attained at the end of the algorithm (again at $r = R = 5000$) for each combination. The results are summarized at the following tables.
From Tables 6 and 7, we observe that for most parameter combinations, all learning sessions conclude with a zero ΔQ_1 for the smaller λ values. However, for $\lambda = 1.05$, the algorithm fails to converge and exhibits a high relative error. Notably, increasing λ from 1.001 to 1.01 generally accelerates convergence, although this is not guaranteed in every case. In one instance, a higher λ (specifically 1.01) leads to a slower average convergence of ΔQ_1 to zero, indicating the importance of initial random shooting choices. We also observe that results for the highest λ are suboptimal, with the algorithm failing to converge in varying numbers of sessions, such as in 1 out of 10 or even 5 out of 10 cases. Notably, when convergence does occur, it happens relatively quickly, with sessions typically completing in under 1000 iterations. These findings also highlight the trade-off between *exploration* and *exploitation*, as discussed earlier.

Finally, in table 8 we see how many learning sessions were run and how many converged to zero error estimate ΔQ_1 for different values of D and λ.

5. Discussion

We proposed an algorithm for estimating the unknown game parameters and the optimal strategies through the course of repeated plays. We tested the algorithm for two models of the kill probability function and found that it converged for the majority of tests. Furthermore, we observed the established relationship between higher learning rates and reduced convergence quality, underscoring the trade-off between learning speed and stability in convergence. Future research could investigate additional models of the accuracy probability function, including scenarios where the two players employ distinct models, and work toward establishing theoretical bounds on the algorithm's probabilistic convergence.

Author Contributions: All authors contributed equally to all parts of this work, namely conceptualization, methodology, software, validation, formal analysis, writing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Polak, Ben. *Backward Induction: Reputation and Duels*. Open Yale Courses. `https://oyc.yale.edu/economics/econ-159/lecture-16`
2. Prisner, Erich. *Game theory through examples*. Vol. 46. American Mathematical Soc. (2014).
3. Fox, Martin, and George S. Kimeldorf. "Noisy duels." *SIAM Journal on Applied Mathematics*, vol.17, pp. 353-361 (1969).
4. Garnaev, Andrey. "Games of Timing." In *Search Games and Other Applications of Game Theory*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 81-120 (2000).
5. Radzik, T. "Games of timing related to distribution of resources." *Journal of optimization theory and applications*, vol. 58, pp. 443-471 (1988).
6. Roy, Dilip. "The discrete normal distribution." *Communications in Statistics-theory and Methods*, vol. 32, pp. 1871-1883 (2003).
7. Ishii, Shin, Wako Yoshida, and Junichiro Yoshimoto. "Control of exploitation–exploration meta-parameter in reinforcement learning." *Neural networks*, vol. 15, pp. 665-687, (2002).
8. Geman, Stuart, and Donald Geman. "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images." *IEEE Trans. on pattern analysis and machine intelligence*, vol. 6, pp. 721-741, (1984).