Available online at www.sciencedirect.com
SCIENCE DIRECT®
Applied Soft Computing xxx (2006) xxx–xxx
Applied Soft Computing
www.elsevier.com/locate/asoc

Chaotic time series prediction for the game, Rock-Paper-Scissors

Franco Salvetti [a,*], Paolo Patelli [b], Simone Nicolo [a]

[a] Department of Computer Science, University of Colorado at Boulder, 430 UCB, Boulder, CO 80309–0430, USA
[b] T-13 Complex System Group, Theoretical Division, CNLS Los Alamos National Laboratory, Mail Stop B-213, Los Alamos, NM 87545, USA

Abstract

Two players of Rock-Paper-Scissors are modeled as adaptive agents which use a reinforcement learning algorithm and exhibit chaotic behavior in terms of trajectories of probability in mixed strategies space. This paper demonstrates that an external super-agent can exploit the behavior of the other players to predict favorable moments to play against one of the other players the symbol suggested by a sub-optimal strategy. This third agent does not affect the learning process of the other two players, whose only goal is to beat each other. The choice of the best moment to play is based on a threshold associated with the Local Lyapunov Exponent or the Entropy, each computed by using the time series of symbols played by one of the other players. A method for automatically adapting such a threshold is presented and evaluated. The results show that these techniques can be used effectively by a super-agent in a game involving adaptive agents that exhibit collective chaotic behavior.

Keywords: Time series prediction; Chaos theory; Game theory; Local Lyapunov Exponent; Entropy filtering; Reinforcement learning; Agent irrationality; Rock paper scissors game

1. Introduction

Game theory [12] and the related concept of Nash equilibrium [11] have produced important practical and theoretical results using the idealization of a *rational agent* [9] which seeks only to maximize its utility. Despite these results, however, it is evident that a rational agent cannot be a completely accurate model of a real agent [14,7]. As a consequence we are observing an increasing interest in the modeling of agent-*irrationality*: ''Standard models in economics stress the role of intelligent agents which maximize utility. However, there may be situations where, for some purposes, constraints imposed by market institutions dominate intelligent agent behavior'' [6]. A possible alternative to rationality is ''bounded-rationality'', where the agent always takes the action that is expected to optimize its performance measure, given some informational and computational constraints [4]. Also, an agent which makes decisions by using simple heuristics or rules of thumb is considered a bounded-rational agent [8].

Real players of Rock-Paper-Scissors[1] (RPS) always use some impulse or inclination to choose a throw, and will therefore settle into unconscious but nonetheless predicable patterns [15]. We expect that such patterns are weaknesses in their behavior.

Chaos in the probability space trajectory has been demonstrated in recent studies of RPS [16,17], where the players change the probability of playing each symbol by using reinforced learning [20] based on coupled Ordinary Differential Equations (ODE). Even with a more *realistic* adaptive agent which uses reinforced learning based on micro-founded heuristics [8] there is still chaos in the probability space trajectories [13]. We conjecture that although local instability on attractors prohibits accurate long-term predictions, short-term predictions can be made with varying degrees of accuracy [10].

Given a time-series which represents the behavior of a system, the usual aim is to predict its behavior in the future [22]. Here we are interested in showing that by means of the Local Lyapunov Exponent (LLE) [1,2] it is possible for a super-agent to predict the behavior of one player [5] of RPS in order to ameliorate its own performance. We consider the game RPS only as a metaphor of, for example, some real market where the super-agent represents a speculator.

Our goal is not to suggest the optimal symbol to be played, but to develop a *theory of the best moment for playing* the symbol suggested by a sub-optimal strategy. We base our analysis on the observable behavior of one of the players, and show that it is possible for the super-agent to improve its performance by playing rounds intermittently on the basis of indicators such as the LLE and the Entropy.

* Corresponding author. Tel.: +1 303 817 9805; fax: +1 303 565 6883.
E-mail addresses: franco.salvetti@colorado.edu (F. Salvetti), paolo@lanl.gov (P. Patelli), simone.nicolo@colorado.edu (S. Nicolo).
[1] See http://www.worldrps.com for an exhaustive explanation of the game.

1568-4946/$ – see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2006.01.006

2. Framework

The RPS game is a two-person, zero-sum game whose payoff matrix is presented in Table 1.

The RPS game has a Nash equilibrium in the mixed strategies space. In the Nash equilibrium each player randomizes the pure strategies with probability 1/3. At equilibrium no player has an incentive to deviate from the Nash strategy.

We study a repeated version of RPS where the same players play the same game iteratively. Unlike a single-shot game, a repeated game allows players to use strategies contingent on past moves by exploiting, if possible, any weakness in the strategy of the other player. Learning and adaptation become fundamental components of the players' interaction.

Our agents use a reinforced learning algorithm to adapt their behavior in response to changes in the behavior of the other player. The reinforcement learning algorithm updates the probability associated with each pure strategy on the basis of the success of past strategies. If the player plays the strategy $s_i \in \{R, P, S\}$, then the complementary strategy for s_i, identified by $\bar{s}_i$, is the strategy that wins against s_i. For example, if the agent plays *Rock*, then the complementary strategy is *Scissors*. If a pure strategy is successful, then the player increases the probability associated with that strategy, and decreases the probability of playing the complementary strategy. Vice-versa, if the strategy is unsuccessful, the agent decreases the probability associated with that strategy and increases the probability of the complementary strategy. The agent updates the probability of playing strategy s_i, $P_{t+1}(s_i)$ at time $t+1$ in the following way:

$$P(s_i)_{t+1} = P(s_i)_t + w_i(\alpha(1 - P(s_i)_t)) \quad (1)$$

where w_i assume value 1 if the strategy outcome is positive, and -1 if negative. The parameter α is called the *learning rate,* and determines the agent's adaptation velocity. We update the probability associated with the complementary strategy in a similar way:

$$P(\bar{s}_i)_{t+1} = P(\bar{s}_i)_t + (1 - w_i)\alpha(1 - P(\bar{s}_i)_t) \quad (2)$$

We perform different experiments changing each agent's initial condition and learning rate parameter α. The projection of an agent's trajectory in the probability simplex for an experiment with $\alpha = 0.1$ and 20,000 repetitions is reported in Fig. 1.

The trajectory resembles a *random* set of points. However, if we project the trajectory within some short intervals of time, as shown in Fig. 2, we notice that the set of seemingly *random* points in Fig. 1 has a precise internal structure, typical of chaotic phenomena. These *scattered orbits* are frequently present in trajectories generated by using sets of heuristics employed to update the probability distribution of each player. The *orbits* in Fig. 2 are not as *smooth* as those proposed in [16,17] because the update of the probability is not controlled by an equation, but follows heuristics which should capture the behavior of a real player who, in a highly discrete way, can change the probability used to play a symbol.

Table 1
Payoff matrix of two-person Rock-Paper-Scissors

Player B	Player A		
	R	*S*	*P*
R	0, 0	1, −1	−1, 1
S	−1, 1	0, 0	1, −1
P	1, −1	−1, 1	0, 0

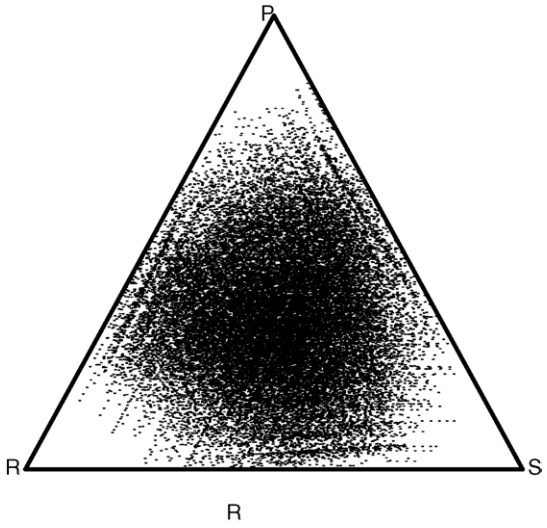

Fig. 1. Trajectory of 20,000 points in the probability simplex generated by two agents learning with $\alpha = 0.1$.

For instance a real player can decide to halve the probability of playing *Rock* as a result of a sequence of losses, updating the other probability to preserve:

$$P_{\text{Rock}} + P_{\text{Paper}} + P_{\text{Scissors}} = 1 \quad (3)$$

Some of these heuristics capture the empirical observation that a player increases its confidence in a symbol after a win, and that the increase in confidence is somehow related with the current probability distribution.

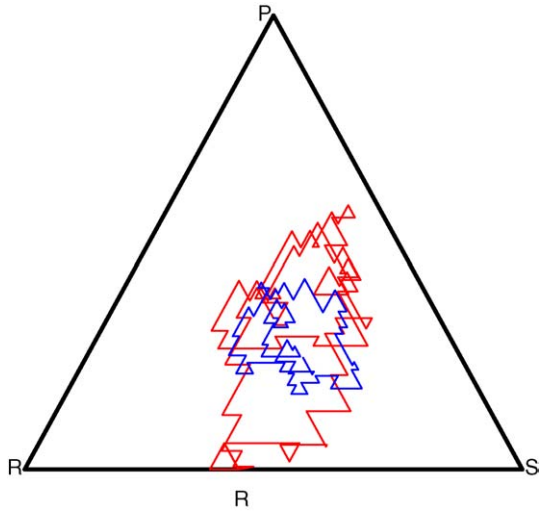

Fig. 2. Two portions of the trajectory of 20,000 points in the probability simplex generated by two agents learning with a probability update of $\alpha = 0.1$ as presented in Fig. 1.

We must notice that in the process of updating their probability distribution, the two players, when they use the same set of heuristics and the same parameters, end up with a mediocre result of nearly one-third wins, one-third losses and one-third ties. Moreover in a long game they play each symbol with the same frequency, namely nearly one-third *Rock*, one-third *Paper* and one-third *Scissors*. Nonetheless, our super-agent, by analyzing their behavior, is able to *predict* the best moment to play.

The results presented in [13] strongly suggest that such a trajectory is chaotic by showing that the Lyapunov Exponent is positive and that the capacity dimension d_c is close to 2. The presence of a Nash Equilibrium in the mixed strategy, namely the uniform distribution $P_{\text{Rock}} = 1/3$, $P_{\text{Paper}} = 1/3$ and $P_{\text{Scissors}} = 1/3$, is not considered attractive by real players who want to win more than one-third of the rounds and avoid losing one-third of the rounds. This *natural* behavior is responsible for this fluctuation in the probability space, and thus for the resulting chaotic trajectories.

2.1. Super-agent goals

Given the situation described in the previous section, the aim of the super-agent is to predict the *right* moment to play against player A, where the best moment is any moment at which the future behavior of player A is more predictable. At such a moment the super-agent plays using a benchmark strategy as described later. One assumption is that the behavior of the super-agent does not affect the learning process of the other two players. This is a reasonable assumption if we think for instance of the currency market, in which the behavior of an individual with small resources will not affect the oscillations of exchange rates.

The super-agent plays against player A in the sense that before seeing the symbols played by players A and B in a match, the super-agent decides what symbol to play against A and auto-evaluates its performance at the moment at which player A actually plays. The aim of the super-agent is to predict the future behavior of player A and to decide what symbol to play in accordance with that prediction.

As noted above, the super-agent is not obliged to play all the rounds. It plays intermittently using the benchmark strategy when the future behavior of player A is considered predictable. Compared with the usual time-series forecasting, the emphasis here is not on the prediction of the future behavior by itself, but on the prediction of the best moment to make a certain prediction.

Other assumptions of the super-agent are that the underlying phenomena are chaotic, and that the Entropy and the LLE are good indicators of the best moment at which to play.

2.2. Techniques employed

The following is a brief description of the techniques and terminology used to develop the super-agent. Considering that the super-agent wants to rule out some rounds in order to improve its performance, we use the term *filtering* to stress the idea of intermittency of play.

2.2.1. Benchmark strategy

In order to evaluate the effectiveness of the *filtering* process, by means of either the LLE or the Entropy, we need a benchmark strategy that will be used when it is the *right* moment to play in order to decide what symbol should be played by the super-agent.

The benchmark strategy always plays the *opposite* symbol of the most frequent one in the *recent past*, where the opposite of the symbol s_i is the symbol $\bar{s}_i$ if and only if $\bar{s}_i$ beats s_i. Considering that we want to look at the *recent past*, we face the problem of determining a reasonable size for a temporal window where we *count* the number of symbols recently played by the player A. At one extreme we can simply look at the most recent symbol, and on the other hand we can consider the whole sequence played so far. The former presents an evident disadvantage, as we have seen, because in the long run both players play with uniform distribution Nash = {1/3, 1/3, 1/3}, and therefore if we look too far into the past, we end up playing almost randomly. From the other side if one looks just at the last symbol played one can find too little information on the current behavior of the player A. One way to determine the *correct size* of the *memory* is by experimentation. We run our player iteratively with all possible sizes of memory, and as we can see in Fig. 3, a peak in the number of wins appears when the memory has a size of around 19.

Considering that this strategy represents only a reasonable benchmark that we want to outperform, we do not spend too much effort in determining the optimality of the memory. However, it is absolutely reasonable to consider, as we do for the adaptive threshold, that the super-agent adjusts, by experience, the size of its memory in a way such that it maximizes its own expected performance. In the experiments with the Entropy we choose a different memory size value, the one that maximizes the number of wins in a simulation based on the sequence of symbols played by the player A.

2.2.2. Local Lyapunov Exponent

Given an initial point x_0 and a point close to it $x_0 + \delta_0$ let δ_n be the separation of the two evolved points after n steps. If $|\delta_n| \approx |\delta_0| e^{n\lambda}$, then λ is called the Lyapunov Exponent (LE). A

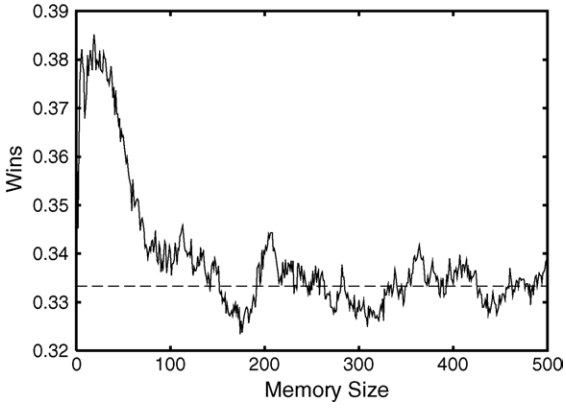

Fig. 3. Graph of the number of wins as function of the size of the memory employed in the benchmark strategy by the super-agent. For a specific sequence of symbols the best size of memory is 19.

positive value of LE is a signature of chaos and it can be seen as a measure of the divergence of nearby trajectories [19]. Therefore, we can consider the LE to be a good indicator of the system *stability* and thus *predictability*.

Given a trajectory in the state-space instead of an ODE which implicitly defines a system, it is possible to compute the LE numerically using, for instance, the Wolf Algorithm [3,23].

The LLE used in this work is a simple and natural extension of the LE and is computed numerically on part of the trajectory rather than the whole. The computation of the LLE by means of the Wolf Algorithm uses $\varepsilon = 0.2$ and searches the nearby trajectory using a cone of vertex angle $\pi/9$ [3]. The idea is that given a trajectory and a discretized time t_i we are interested in computing an LLE related with such a time t_i. We define a time-window of a certain length w and compute the LE by considering only the trajectory in the temporal interval $[t_{i-w}, t_i]$. We also consider such a value to be the LLE at the time t_i.

Using a discrete time we can compute the LLE for the time t_{i+1}, as described before, by simply sliding the temporal window one step further.

2.2.3. Entropy

Given a source $S = \{s_1, \ldots, s_n\}$ and the related distribution of probability $\{p_1, \ldots, p_n\}$, the information [18] associated with an event $E \subseteq S$ is defined as:

$$I(E) = \log \frac{1}{P(E)} \tag{4}$$

The Entropy H of S is defined as the expected value of the information:

$$H(S) = \sum_{s \in S} P(\{s\}) \log \frac{1}{P(\{s\})} \tag{5}$$

The Entropy can be seen as a measure of the predictability of the source. We use the Entropy of a player as an alternative measure for deciding upon the best moment to play.

2.2.4. Embedding

Embedding is a common technique used for time-series prediction in order to *rebuild* the original state-space trajectory of the system [22]. Given a temporal series $y(t)$ coming from experimental measures of a system with dimension d, it is possible to prove [21] that if $m \geq 2d + 1$ the trajectory $(y(t), y(t + \tau), \ldots, y(t + (m - 1)\tau))$ in the reconstructed space is *diffeomorphic*[2] to the actual one, giving us some insight into the system.

3. Experiments

In order to perform our predictions we use the time-series in Table 2 and the probability simplex in Fig. 1.

[2] Diffeomorphic can be thought of as meaning "having the same topology".

Table 2
Temporal sequence of 20,000 symbols played by the player A

t	Symbol
1	*R*
2	*P*
3	*S*
4	*P*
5	*P*
⋮	⋮
20,000	*R*

3.1. LLE filtering

We cannot assume that the super-agent knows the probability distribution P_t used by player A, but we can assume that it is possible to make an estimate of it based on the sequence of symbols played by player A. However, the best estimation of P_t is P_t itself, and hence, as a proof of the concept, we want to demonstrate that a filtering based on the LLE computed on the trajectory P_t improves the performance of the super-agent. With such a positive result the problem is to determine a good estimate of P_t. Knowing the P_t of player A the LLE is computed as described before and the result presented in Fig. 4.

With the LLE computed on the trajectory in the probability space we can *decide* whether to play or not based on its current value. We expect that a low value of the LLE corresponds to a steadier behavior of player A and conjecture that the moments in time associated with such low LLEs are the best moments to play the benchmark strategy. Contrary to our expectations, the actual experiments suggest exactly the opposite, namely that we have an improvement in the performance of the super-agent if it plays when the LLE is above a certain threshold. In Fig. 5, we can appreciate that when the super-agent plays all the rounds (i.e. the left-hand side of the graph) the benchmark strategy outperforms one-third, which means that the benchmark strategy performs better than a pure Nash strategy. If we increase the threshold, forcing the super-agent to play

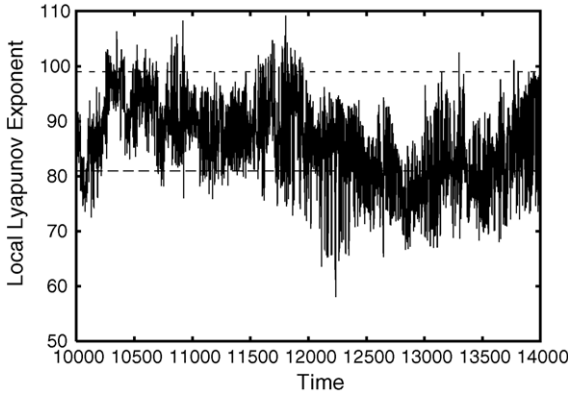

Fig. 4. Graph of the LLE computed for a trajectory in the probability state space of length 20,000, with a sliding window of length 10,000 and increment 1, computed with the Wolf algorithm which employs a cone of vertex angle of $\pi/9$. The two horizontal lines represent the *optimal* interval where we want to play.

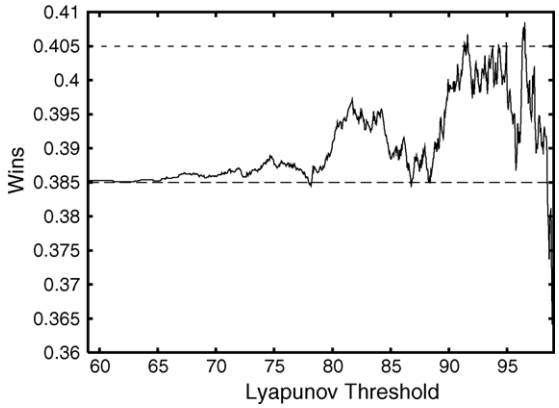

Fig. 5. Given the LLE in Fig. 4 and the relative sequence of symbols played by player A, we play the benchmark strategy with memory length 19 when the LLE is greater than the threshold. The percentage of wins is reported for an increasingly high value of the threshold. On the left-hand side of the graph the value 0.385 represents our base-line, namely the fraction of wins when we play each round (no threshold) with the benchmark strategy.

intermittently, the performance improves in terms of percentage of wins.

In Fig. 5, we can see that for values of the threshold in the interval [82,98] there is an increase in performance, which sometimes reaches values of roughly 40%, that is, nearly 2% more with respect to the baseline of the benchmark strategy, which wins 38.5% of the time. This is the first evidence that playing intermittently based on the LLE improves the performance. Different experiments with different window lengths for the LLE gave different degrees of improvement. So far it appears that a window length of 10,000 works better than shorter lengths such as 1000, but we will see that with small window sizes, such as 100, it is possible to get even better results. Given these positive results we are now interested in determining the value of the threshold automatically; the next section will address this problem.

3.1.1. Adaptive threshold

As seen before, there are values of the threshold that give improvement in the performance. Here we want to show that it is possible to conceive an algorithm to determine a *good* value of the threshold automatically. The goal is to have a method which can determine good values of the threshold both on the LLE and on the Entropy.

It is intuitive that an adaptive value of the threshold should be computed based on the observed values of the LLE. Considering that values too far in the past are in general not informative with respect to the current behavior, we base the computation of the Adaptive Threshold (AT) on the most recent values available in a window of size w. The mean of the LLE in this window can be used as good estimation of the current behavior of the LLE. Considering that we are playing when the LLE is above a certain threshold, a conservative approach is to increase the mean of an amount that can be computed based on the current variability of the LLE. Considering that the standard deviation is a common way to measure risk and variability, we use an estimate of the standard deviation computed on the observations.

Given a window of size w and a temporal series y_t (e.g., LLE) we look into the past values from the current point in time and then compute the mean:

$$\mu_t = \frac{1}{w}\sum_{i=0}^{w-1} y_t \tag{6}$$

and the standard deviation of the sample:

$$\sigma_t = \sqrt{\frac{\sum_{i=0}^{w-1}(y_t - \mu_t)^2}{w}} \tag{7}$$

and we conjecture that a good threshold for any point in time is:

$$T_t = \mu_t + \sigma_t \tag{8}$$

Given the LLE series in Fig. 4, we can apply Eq. (8) and compute the value of the threshold at any point in time. The results are reported in Fig. 6.

As already noted good values of the threshold should be in [82,98]. The graph of the adaptive threshold in Fig. 6 is consistent with our expectations because the automatically computed AT is in the interval [82,98]. The overall performance of the super-agent playing the benchmark strategy whenever the LLE is above the AT reaches a win percentage of 39.7. Considering that when the benchmark strategy is played each time, we reach a win percentage of 38.5, we observe that playing intermittently with the AT outperforms the base-line strategy. Furthermore, such a result is comparable in magnitude to the average of the percentage of wins in the interval of thresholds [82,98] reported in Fig. 5.

By using Eq. (8) for computing the AT on the Entropy series, numerically computed on the sequence of symbols thrown by player A, we obtain values of AT in the interval of threshold values where there is improvement. The *good* values of the threshold were determined by visual inspection on the Entropy graph of Fig. 11.

The fact that Eq. (8) produces *good* AT if applied to both the Entropy and LLE series increases the confidence in the robustness of this method.

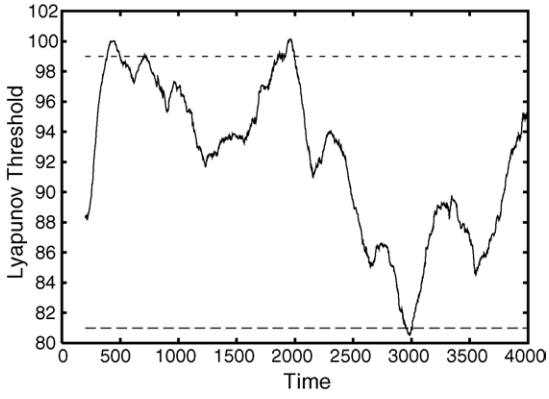

Fig. 6. Adaptive LLE threshold computed on the LLE graph of Fig. 4 by using (8), with a sliding window of length $w = 200$. The two horizontal lines represent the *optimal* interval where an increase in the percentage of wins is expected.

3.2. Estimated probability trajectory

Given the positive results presented in the previous section, we move the problem from predicting the best moment to play to estimating P_t given the time-series in Table 2. A simple approach that gives fairly accurate results is to use a sliding window of length w which moves along the symbol series in steps of length 1, and to estimate P_t as the frequency of each symbol in such a window. Using this technique and the data reported in Table 2 it is possible to compute an estimate of the probability distribution P_t as reported in Fig. 7. If we compare Fig. 7, the estimated P_t, with Fig. 1, the real P_t from which the symbols in Table 2 are derived, we may note the expected difference.

The value of w must be chosen carefully because it dramatically affects the estimate P_t, and consequently the computation of the LLE based on the estimated trajectory in the probability space of P_t itself. In order to understand the effect of w on the estimate P_t we should notice that when w is extremely small, the number of possible combinations of the three symbols $\{R, P, S\}$ is relatively small. For instance when $w = 1$, we are observing only one symbol in the past, and hence the estimate P_t can assume only three possible values in the probability space corresponding to the points (1, 0, 0), (0, 1, 0) and (0, 0, 1). Any resulting trajectory would be between three such points and give little information on the current predictability of the phenomenon. At the other extreme of the spectrum is the case in which w is too large. For instance, when $w = 10,000$, by the fact that in the long run each player plays each symbol with frequency very close to one-third, the resulting estimated trajectory is a sequence of points in a ball centered in (1/3, 1/3, 1/3) of radius $\varepsilon > 0$ where ε gets smaller for increasingly high values of w. Practically speaking, high values of w collapse the estimated trajectory at point (1/3, 1/3, 1/3) giving no information on the predictability of the phenomenon. In these experiments w is empirically determined as a value able to generate a trajectory in balance between collapsed and high discrete.

Using the reconstructed trajectory presented in Fig. 7 we can compute the LLE as shown in Fig. 8.

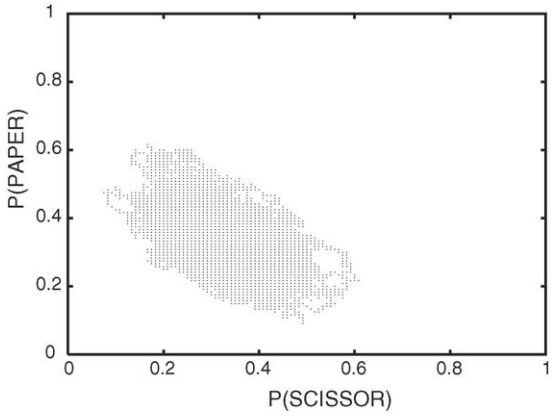

Fig. 7. Projection in the plane P(SCISSOR) $\times$ P(PAPER) of the estimated trajectory in the probability space generated with $w = 120$ from the 20,000 symbols s played by the player A in Table 2.

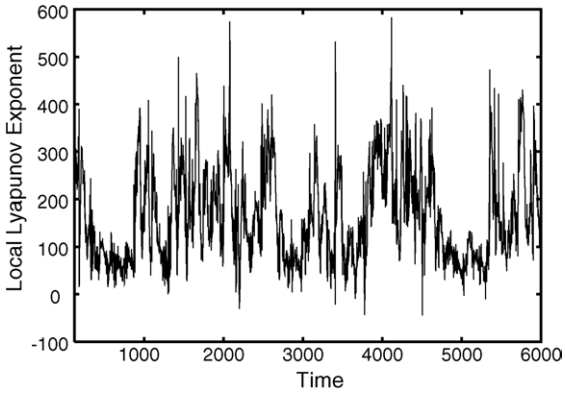

Fig. 8. Graph of the LLE computed for the reconstructed trajectory in the probability space shown in Fig. 7, with a sliding window of length 120 and increment 1, computed with the Wolf algorithm which employs a cone of vertex angle $\pi/9$ and $\varepsilon = 0.01$.

When we compute the LLE on the original state space trajectory, we use a window of size $w = 10,000$ and achieve an unexpected improvement in the performance of the benchmark strategy by playing when the current value of the LLE is above the threshold. This is in contrast with our expectation, because we argue that low values of the LLE correspond with the stability and therefore the predictability of the system. In this experiment we drastically reduce the window length to $w = 50$. At this point playing when the current LLE is less than the threshold we have the expected improvement in the performance of the super-agent. In Fig. 9, we observe that the number of wins as a function of the different threshold is better than the results in Fig. 5.

3.3. Entropy filtering

As mentioned previously, the Entropy $H(S)$ is a good indicator of the variability of a source of symbols. The idea here is to use a sliding window of length w which moves with steps of length 1 along the sequence of symbols in Table 2 and to

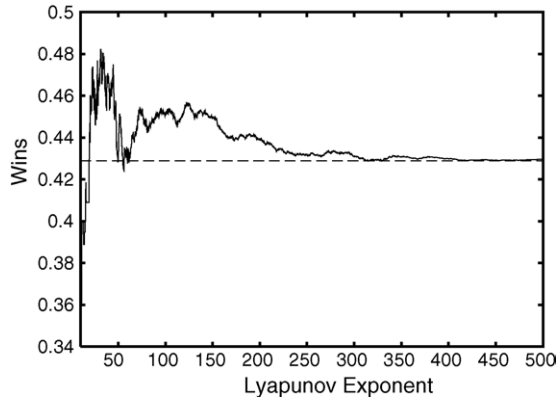

Fig. 9. Given the LLE in Fig. 8 and the relative sequence of symbols played by player A, we play using the benchmark strategy with memory 19 when the LLE is less than the threshold. The number of wins is here reported for an increasingly high value of the threshold. On the right-hand side of the graph, 0.43 represents our base-line, namely the fraction of wins when we play each round.

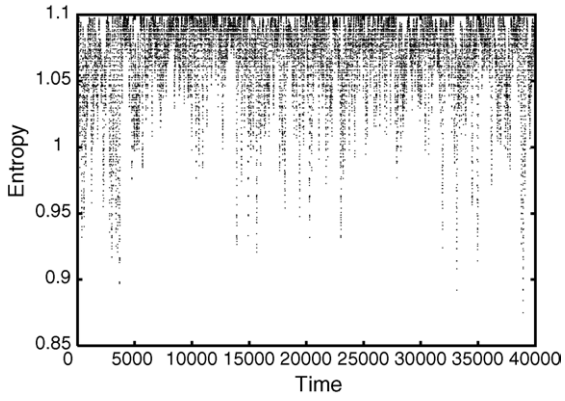

Fig. 10. Entropy computed using the formula (11), with a sliding window of length $w = 100$ along the sequence of 20,000 symbols played by the player A.

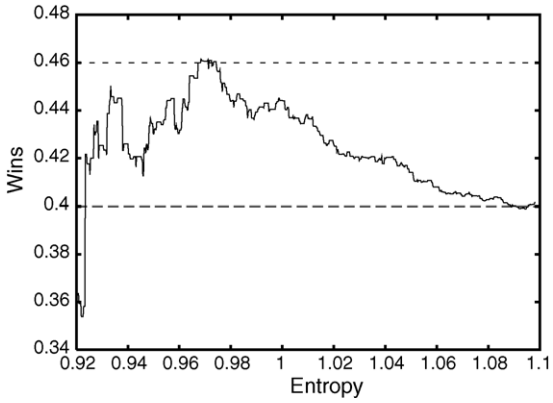

Fig. 11. Given the Entropy in Fig. 10 and the relative sequence of symbols played by player A, we play using the benchmark strategy with memory 28 when the Entropy is less than the threshold. The number of wins is here reported for an increasingly high value of the threshold. On the right of the graph 0.4 represents our base-line, namely the fraction of wins when we play each round.

evaluate the Entropy in such a window by counting each symbol:

$$c_t(x) = \sum_{i=0}^{w-1} (s_{t-i} == x) \quad \forall x \in \{R, P, S\} \tag{9}$$

Next we estimate the probability of each symbol from the probability:

$$p_t(x) = \frac{c_t(x)}{w} \quad \forall x \in \{R, P, S\} \tag{10}$$

and finally we compute the current value of the Entropy:

$$H_t = \sum_{x \in \{R,P,S\}} p_t(x) \log \frac{1}{p_t(x)} \tag{11}$$

The resulting values of H_t for the sequence of symbols played by player A is reported in Fig. 10.

To decide whether the super-agent should play or not at a given time t we use the value of the Entropy H_t at that time. If the value is below a certain threshold, suggesting that the source is stable, the super-agent plays the benchmark strategy. The performance of the super-agent for different values of the threshold of the Entropy is reported in Fig. 11.

The right-hand side of the graph in Fig. 11 shows a performance of 40% when the super-agent plays all the rounds using the benchmark strategy. This value is different from that obtained with the LLE because the benchmark strategy is here based on a different size of memory, and because the sliding window used to estimate the H_t has a different length from the one used to estimate the probability distribution P_t, and therefore the first two available values respectively for $P_{t'}$ and $H_{t'}$ are such that $t \neq t'$. Consequently the numbers of rounds played in the simulation with the LLE and the Entropy are different. Considering that we are interested in comparing the capability of this technique in boosting a benchmark strategy and not in the performance of the benchmark strategy per se, this difference between the baselines of the two methods is not considered relevant.

If we compare the graphs in Figs. 11 and 9 we note that playing by using the Entropy-based filtering increases the performance of the benchmark strategy more than playing by using the filtering based on the LLE. Moreover the interval of values of the threshold in which this improvement appears is larger, and hence the Entropy-based filtering steadier and more reliable.

As previously indicated, the real super-agent must fix an a priori threshold in order to decide whether to play.

For the Entropy we use the same technique presented in the case of the LLE, meant to adapt the threshold automatically on the basis of the available observations. This gives similar results in terms of improvement. The percentage of wins with adaptive threshold is higher than the benchmark strategy, but lower than the possible best value, which would have been reached with a threshold of 0.97.

3.3.1. Sequence of symbols, Nash player

In order to show that the improvement provided by the Entropy-based filtering is related with the actual *predictable* behavior of player A, we generate a sequence of random symbols extracted from a uniform distribution. We compute the Entropy of such a sequence of symbols and force the super-agent to play against the Nash player.[3] The resulting performances for different values of the threshold are reported in Fig. 12.

On the right of the graph 0.33 represents our base-line, namely the fraction of wins when we play each round. On the right-hand side of the graph in Fig. 12, the performance of 33% represents the best that any strategy can do against a random player. As expected there is no improvement in the performance of the super-agent when it plays against a random player. This observation increases our confidence that there is a structure in the behavior of player A.

[3] There is no strategy that can outperform one-third wins against a random player.

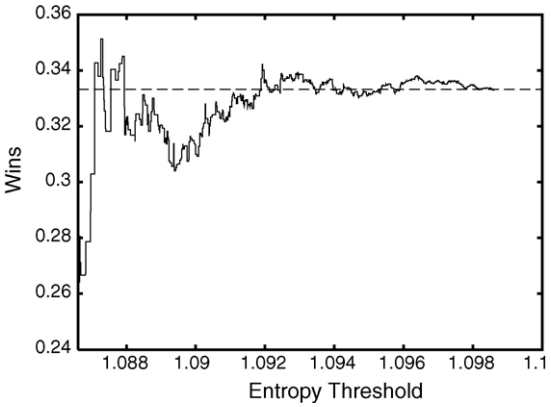

Fig. 12. Given a sequence of 50,000 symbols played by a Nash player, and the Entropy computed with the formula (11) with $w = 100$, we play using the benchmark strategy with memory 28 when the Entropy is lower than the threshold. The number of wins is here reported for an increasingly high value of the threshold.

3.4. Embedded entropy

Using a filtering method based on the Entropy, we can also increase the performance of our super-agent. We explore the opportunity of embedding the temporal sequence of symbols in Table 2 in order to reconstruct the possible *original* probability space. As it is difficult to conceive a way to determine the *correct* value for the number of dimensions, an arbitrary but reasonable value of $m = 5$ is chosen. We conjecture that a relatively small number of dimensions should be sufficient for capturing the essential aspects of the underlying phenomenon. Symbols far apart in the temporal series are expected to be unrelated, hence a value of $\tau = 2$ is chosen in this experiment.

In order to compute the Embedded Entropy we consider a sliding window of size w on the sequence of symbols looking into the past from the last observed symbol at time t. Within such a window, we count the occurrences of any possible sequence $\underline{x} \in \{R, P, S\}^d$ by considering

$$\underline{s}_t = (s_t, s_{t-\tau}, ..., s_t(m-1)\tau) : c_t(\underline{x}) = \sum_{i=0}^{w-(m-1)\tau} (\underline{s}_{t-i} == \underline{x}) \tag{12}$$

from which we can estimate the probability $p(\underline{x})$ as in Eq. (10) and the Entropy H_t as we did in the Eq. (11). With the current value of the Entropy we can decide whether to play using the benchmark strategy or not if H_t is below a certain threshold. Fig. 13 shows the performance of the super-agent for different values of the threshold.

In Fig. 13, we should note that the base-line of wins is 40% and that there is an improvement of nearly 4%, which is less than the 6% attained with the Entropy-based filtering. It is important to point out that even if there is a decrease in overall performance, this method seems more robust in terms of the consistency in the improvements for different values of the threshold.

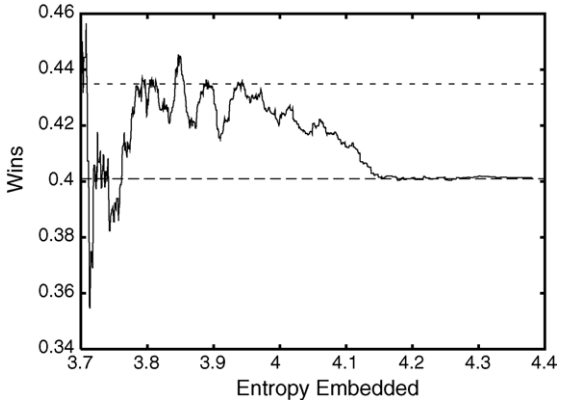

Fig. 13. Given the Embedded Entropy H_t, with $\tau = 2$, $m = 5$, $w = 100$ and the relative sequence of symbols played by player A, we play using the benchmark strategy with memory 28 when the Embedded Entropy is lower than the threshold. The number of wins is here reported for an increasingly high value of the threshold. On the right of the graph 0.4 represents our base-line, namely the fraction of wins when we play each round.

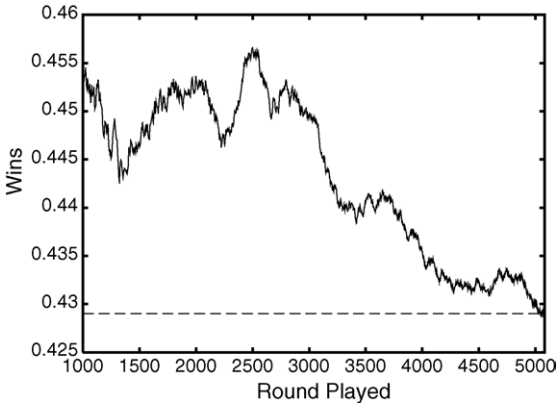

Fig. 14. Graph that relates the number of wins with the number of rounds played.

4. Trade-off rounds/wins

The size of the range of effective thresholds is crucial because it is always difficult to predict a good threshold value, and because the larger it is the higher the number of times the super-agent plays. We are interested not only in improvement in performance but also in having the super-agent play successfully as many times as possible in order to maximize the total number of wins. As we can see in Fig. 14, the more frequently we make the super-agent play, the closer the performance is to the benchmark strategy.

It is important to note that it is possible to play nearly half of the time and still have an improvement of more than 2% over the benchmark strategy.

5. Conclusions

This research demonstrates that the chaotic nature of the trajectory in the probability space of a player of Rock-Paper-Scissors, along with its time-series of symbols whose generation

is based on micro-founded heuristics, can be a weakness of that player. In fact, such a trajectory contains enough structure to be used by a super-agent to predict its future behavior.

The framework of this repeated version of the game is meant to represent the more general problem of prediction in the presence of chaos (e.g., the currency market). For instance, it would be possible to increase the discretization of a stock-market temporal series, transforming it into a sequence of symbols in {*up*, *down*, *steady*}. The methods presented in this paper may then be employed for choosing the moments at which sub-optimal prediction of the actual trends is more likely to be correct.

Our preliminary results are encouraging and show that by accepting intermittency in playing it is possible to decide the best moment to play and consequently to improve the performance of a simple and naïve strategy. Filtering based on the Entropy appears more effective than that based on the LLE, and embedding the trajectory in order to compute the Entropy slightly degrades performance while increasing robustness. The effectiveness of a method for automatically adapting the threshold is proven to be effective.

Many of the algorithms used and implemented here employ large numbers of parameters, each of which impacts the final performance. A study of the best values of such parameters could reveal unseen relations among them. Experiments with different benchmark strategies and different temporal series are important to determine the robustness of the different methods.

References

[1] H.D.I. Abarbanel, R. Brown, M.B. Kennel, Local Lyapunov Exponents from observed data, J. Nonlinear Sci. 2 (1992) 343–365.
[2] W.A. Barnett, A.P. Kirman, M. Salmon, Local Lyapunov Exponents: predictability depends on where you are, Nonlinear Dynam. Econ. (1996).
[3] L. Bradley, Wolf's algorithm for computing Lyapunov Exponent from data, Technical Report Chaotic Dynamics–CSCI 6446, University of Colorado at Boulder, 2004.
[4] E. Eberbach, $-Calculus Bounded Rationality = Process Algebra + Anytime Algorithms, Appl. Math.: Perspect. Challenges (2001) 213–220.
[5] I. Erev, A. Roth, Predicting how people play games: reinforcement learning in experimental games with unique, mixed strategy equilibrium, Am. Econ. Rev. (1999).
[6] J.D. Farmer, P. Patelli, I.I. Zovko, The predictive power of zero intelligence in financial markets, Nature 24 (2003) 2003.
[7] G. Gigerenzer, R. Selten, Bounded Rationality: The Adaptive Toolbox, MIT Press, Cambridge, Mass, 2002, pp. 13–36.
[8] G. Gigerenzer, P.M. Todd, ABC Research Group, Simple Heuristics that Make us Smart, Oxford University Press, 2000.
[9] E. Kalai, E. Lehrer, Rational learning leads to Nash equilibrium, Econometrica 61 (1993) 1019–1045.
[10] J. McNames, J.A.K. Suykens, J.J. Vande-walle, Winning entry of the K.U. Leuven time series prediction competition, Int. J. Bifurcation Chaos 8 (9) (1999) 1485–1500.
[11] R.B. Myerson, Nash equilibrium and the history of economic theory, J. Econ. Literature 37 (1999) 1067–1082.
[12] J.V. Neumann, O. Morgenstern, Theory of Games and Economic Behavior, Princeton University Press, 1944.
[13] S. Nicolo. Chaos and Game Theory, a study on Rock-Scissors-Paper. Technical Report Chaotic Dynamics-CSCI 6446, University of Colorado at Boulder, 2004.
[14] S. Pearlstein. Two Americans to share Nobel prize in economics. Washington Post, October 10, 2002.
[15] World RPS. Advanced RPS. Technical report, World Rock Paper Scissors Society, 2004.
[16] Y. Sato, E. Akiyama, J.D. Farmer, Chaos in learning a simple two person game, Proc. Natl. Acad. Sci. USA 99 (2002) 4748–4751.
[17] Y. Sato, J.P. Crutchfield, Coupled replicator equations for the dynamics of learning in multi-agent systems, Phys. Rev. E 67 (1) (2003) 40–43.
[18] C.E. Shannon. A mathematical theory of communication. Bell Syst. Tech. J. 27 (1948) 379–423 and 623–656.
[19] S.H. Strogatz, Nonlinear Dynamics and Chaos: with Applications to Physics, Biology, Chemistry, and Physics, Westview Press, Perseus Book Group, 1994.
[20] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, The MIT Press, 1998.
[21] F. Takens, Detecting strange attractors in fluid turbolence, Dynam. Syst. Turbolence (1981).
[22] A.S. Weigend, N.A. Gershenfeld, Time Series Prediction, Addison Wesley, 1992.
[23] A. Wolf, J.B. Swift, H.L. Swinney, J.A. Vastano, Determining Lyapunov Exponents from a time series, Physica D 16 (1985) 285–317.