Deep Learning Based Game-Theoretical Approach to Evade Jamming Attacks

Sandamal Weerasinghe[1(✉)], Tansu Alpcan[1], Sarah M. Erfani[1], Christopher Leckie[1], Peyam Pourbeik[2], and Jack Riddle[3]

[1] Melbourne School of Engineering, The University of Melbourne, Melbourne, Australia
pweerasinghe@student.unimelb.edu.au
[2] Defence Science and Technology Group, Canberra, Australia
[3] Northrop Grumman Corporation, Falls Church, USA

Abstract. Software-defined radios (SDRs) with substantial cognitive (computing) and networking capabilities provide an opportunity for malicious individuals to jam the communications of other legitimate users. Channel hopping is a well known anti-jamming tactic used in order to evade jamming attacks. We model the interaction between a transmitter, who uses chaotic pseudo-random patterns for channel hopping, and a sophisticated jammer, who uses advanced machine learning algorithms to predict the transmitter's frequency hopping patterns as a non-cooperative security game. We investigate the effectiveness of adversarial distortions in such a scenario to support the anti-jamming efforts by deceiving the jammer's learning algorithms. The optimal strategies in the formulated game indicate how adversarial distortions should be used by the players at every step of the game in order improve their outcomes. The studied jamming/anti-jamming scenario combines chaotic time series generators, game theory, and online deep learning.

Keywords: Jamming · Game theory · Adversarial learning

1 Introduction

Recent advances in software-defined radios (SDRs), cognitive networking technologies, and the increasing availability of low-cost hardware, have resulted in most applications becoming dependent on wireless networks for their regular operations. Inevitably, this has given adversaries new opportunities to conduct attacks and harm systems that rely on wireless networks. One of the most common forms of attacks in wireless networks is jamming attacks. Adversaries can utilize cheap and compact transmitters and receivers [8] to scan the transmission channels in a particular area and disrupt the communication between two or more legitimate parties by causing interference or collisions at the receiver side.

This work was supported in part by the Australian Research Council Discovery Project under Grant DP140100819 and by the Northrop Grumman Corporation.

L. Bushnell et al. (Eds.): GameSec 2018, LNCS 11199, pp. 386–397, 2018.
https://doi.org/10.1007/978-3-030-01554-1_22

This paper focuses on a particular scenario where stochastic channel hopping is utilized as an evasion strategy in the presence of a sophisticated jammer. Consider a unit immobilized in a contested environment with its location unknown to friendly search and rescue units in the area. Assume that the immobilized unit (T) still has the capability to transmit signals using its radio transmitter. Multiple friendly units in the area (receivers R) attempt to locate and rescue the immobilized unit (T) by using radio direction finding (RDF). In order to successfully triangulate the location of T, the search and rescue units need to establish communication with it via one of n pre-defined channels. All units are equipped with software defined radios (SDRs), which allows them to switch the transmission/receiver channels on the fly.

The problem of locating the transmitter T becomes challenging if an intelligent jammer J attempts to disrupt the communication through interference. Anti-jamming tactics for wireless networks have been an area of research for some time. For example, [13] propose channel surfing/hopping and spatial retreats as possible strategies for evading jamming attacks. In channel hopping, the signal transmitter would proactively switch the transmission channel according to some pattern (shared with the receiver).

In this paper, we propose a scheme in which the transmitter uses chaotic pseudo random patterns for frequency hopping and the jammer attempts to learn the underlying patterns by training a state of the art Recurrent Neural Network (RNN) in an online setting (i.e., updating the prediction model as new data becomes available). This allows the transmitter to deceive the jammer into learning a false representation of the generator functions by introducing adversarial distortions on top of the generated probability distributions. We apply a game theoretic approach to the interaction between a transmitter and a sophisticated jammer where the two players attempt to mislead each other by maliciously distorting data used by the other player to make decisions. The combination of games, pseudo-random generators and deep learning have not been used in jamming applications to the best of our knowledge.

The main contributions of the paper are highlighted as follows:

- A novel adversarial (deep) learning approach to jamming, where the transmitter and jammer attempt to mislead each other by maliciously distorting data used by the other player to make decisions.
- Integrated use of pseudo-random generators and online machine (deep) learning methods in the context of a frequency hopping scheme.
- An overarching security game modeling the decision making in the context of adversarial (deep) learning and deception by the transmitter and receiver.

2 Related Work

We focus on prior work that utilizes game theory for analyzing jamming problems. For example, the paper [3] uses a game-theoretical approach to analyze jamming/anti-jamming behavior between cognitive radio systems. The authors formulate the jamming problem as a fictitious play between the jammer and the

transmitter. In [14], a non-cooperative game is formulated to model the interaction between wireless users and a malicious node that can act as a jammer and an eavesdropper. The authors also utilize a fictitious play based algorithm to find mixed strategy Nash equilibrium solutions. While machine learning techniques have been extensively used for intrusion detection, their usage in jamming applications is relatively recent. For example, [12] uses supervised learning to detect jamming attacks on IEEE 802.11 networks. They train a random forest classification model based on metrics collected from simulations and attempt to predict if a network is under a jamming attack.

In [2], the authors explore the use of chaotic systems to generate frequency hopping sequences. More recently, [6] proposes a pseudo-random sequence generator based on the Chen chaotic system. In order to make multi-step predictions on chaotic time series, [4] propose a variant of RNNs, which attempts to reduce the number of redundant connections between nodes prior to training. More recently, particular interest has been put into predicting chaotic time series using Long Short-Term Memory networks (LSTMs). A knowledge-based prediction modeled combined with a machine learning based prediction model is utilized by [11] to predict chaotic systems.

3 Problem Definition

We consider the interaction between a sophisticated jammer and a transmitter, where the transmitter T is attempting to establish communication with several receivers R while being interfered by the jammer J. At the network level, the only action available to T (J) is choosing a single channel out of the n available channels to transmit (cause interference on). The jammer may successfully disrupt the communication between the transmitter and the receivers by decreasing the *signal-to-interference-plus-noise* ratio (SINR) at the receivers. However, this would then give away the location of the jammer, which may be an unacceptable risk.

Instead of using just one specific channel to transmit signals over a specific time interval, the transmitter switches between the available channels according to some probability distribution p^T. If the transmitter chooses a static probability distribution p^T throughout, the jammer can approximate p^T by observing the channels used by T over a specific time period. To prevent the jammer from approximating p^T, and thereby successfully carry out jamming attacks, the transmitter should periodically change p^T. The manner in which $p^T(k)$ is changed over time cannot be purely random, as that would make it very difficult for the receivers to listen to the correct channel at any given time in order to successfully locate T. But if the changing mechanism is easily perceived, the jammer would learn the new $p^T(k)$ with minimal effort. Therefore, the transmitter utilizes a pseudo-random number generator function $g_i(k) : \mathbb{R} \rightarrow \mathbb{R}$ for each channel $c_i, i \in [1...n]$ in order to create probability distributions that change periodically. In this paper, we use chaotic pseudo-random number generators as a starting point, similar to [2,6]. Then, the probability of selecting channel c_i during the k^{th} interval is given by

$$p_i^T(k) = \frac{g_i(k)}{\sum_{j=1}^{n} g_j(k)}. \quad (1)$$

Making the probability distribution over the n channels during the k^{th} interval $p^T(k) = [p_1^T(k), p_2^T(k), \ldots, p_n^T(k)]$. Note that the transmitter T has the freedom to decide the duration of each time interval in which a particular $p^T(k)$ is used. If the interval duration changes with each $k \in \mathbb{Z}_{\geq 0}$, the jammer can use a *change point detection* algorithm to identify that the probability distribution has changed and react accordingly [7,9]. Therefore, without loss of generality, we assume that the time interval during which a particular probability distribution is used is fixed.

The jammer can observe the channel usage of the transmitter and approximate the probability distribution during the k^{th} interval by creating a histogram ($\hat{p}^T(k)$). In order to successfully interrupt T's transmission, the jammer would have to know in advance the probability distribution T would use in the next time interval. Therefore, the jammer attempts to learn $g_i, i \in [1, n]$ by training prediction models for each g_i as new observations $\hat{p}^T(k)$ become available. For example, by using the observed $\hat{p}^T$ values from intervals 1 to $k-1$, the jammer would be able to predict the possible $p^T(k)$ distribution.

The transmitter, who also has spectrum sensing capabilities, observes the channels utilized by the jammer during the k^{th} interval and attempts to approximate the probability distribution J uses over the n-channels for jamming ($\hat{p}^J(k)$). If the observed $\hat{p}^J(k)$ does not diverge from $p^T(k)$ significantly (i.e., $\hat{p}^J(k) \approx p^T(k)$), it implies that the jammer has closely predicted $p^T(k)$ based on the transmitter's previous probability distributions. To hinder the online learners of the jammer, the transmitter T can mislead the jammer by introducing adversarial distortions to each $p_i^T(k)$. By adding adversarial distortions to each of the probabilities, the transmitter expects the jammer to learn a prediction model that is different from the actual generator functions used by the transmitter and predict a strategy $p^J(k)$ that is significantly different from $p^T(k)$. But the transmitter cannot greedily add significantly large distortions as the receivers who are attempting to locate the transmitter would be unaware of these distortion functions, and would continue to use the original pseudo-random generator functions to decide the channels they would listen to. As the transmitter's main objective is to be located without delay, adding adversarial distortions would have a detrimental effect.

If the jammer successfully predicts the p^T values over a period of time, the transmitter can react by either increasing the adversarial distortion intensity or by switching to different generator functions. If the transmitter switches the generator functions, the jammer would have to restart the learning processes. If the transmitter increases the adversarial distortions, it would make learning the true generator functions harder. Therefore, to prevent the transmitter from deviating from the usual generator patterns, the jammer could periodically add adversarial distortions to its own strategies p^J in order to mislead the transmitter into believing that the jammer has not learned the generator functions.

Therefore, the interaction between that jammer and transmitter has a two way obfuscation nature where both players attempt to confuse each other.

The following specific assumptions are made to reduce the complexity of the transmitter-jammer interaction as a starting point and formalize it as a game:

- The jammer and transmitter will only utilize a single channel at a time (with maximum power to maximize the range).
- The n channels are non-overlapping, therefore jamming on channel c_i would not cause interference on channel c_j where $i \neq j$.
- Jamming is modeled as a discrete event, it will either completely disrupt the communications or not.
- The transmitter has the flexibility of choosing different generators for each channel c_i. But once transmissions begin, the generator assignments are assumed to be fixed.
- While the transmitter can also decide the duration of the time interval to keep a particular probability distribution (i.e., $\Delta t \in [T_1, T_2], \Delta t \in \mathbb{Z}_{\geq 0}$), we assume the duration, in seconds, to be fixed for every time interval.

4 Methodology

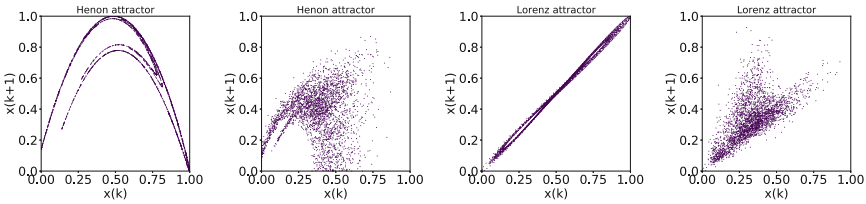
The observations of the players, $\hat{p}^T$ and $\hat{p}^J$, are a combination of the other players' genuine probability distribution, p^J and p^T, and their respective adversarial distortions, d^T and d^J. As introducing adversarial distortions can lead to probability values becoming negative or greater than one, the resulting vector is projected onto the probability simplex as $\hat{p}_i^T(k) = [p_i^T(k) + d_i^T(k)]_P$, where $[.]_P$ is the projection onto the probability simplex Δ defined as

$$\Delta := \left\{ p \in \mathbb{R}^n : \sum_{i=0}^{n} p_i = 1 \text{ and } 0 \leq p_i \leq 1, \ \ \forall i \right\}. \tag{2}$$

4.1 Chaotic Time Series

While hardware based random number generators are available, especially for non-civilian usage, pseudo-random generators are essential in the above scenario for there needs to be a synchronization method (through pre-shared information) between the transmitter and the listeners. As a starting point [2,6], we select several chaotic time series, such as Rossler attractor, Lorenz attractor and Henon attractor as the generator functions for the transmitter. Even though chaotic time series appear to unpredictable and show divergent behavior, they are governed by well-defined nonlinear equations [1]. We will investigate alternative, cryptographic pseudo-random number generators and compare them to chaotic ones in our future work.

At every step of the game, we obtain the corresponding time series values from each generator function and derive the probability distribution of the transmitter by normalizing the values (1). Figure 1 shows the phase diagrams, where



(a) original Henon (b) normalized (c) original Lorenz (d) normalized

Fig. 1. The original phase graphs of the Henon and Lorenz attractors and the decorrelated graphs after combining with the other time series.

the times series value at time $k+1$ is plotted against the time series value at time k, for Henon and Lorenz attractors. The phase diagrams indicate that the normalization process, which creates a dependency among the chaotic attractors, makes each time series decorrelated. Therefore each time series exhibits more "randomness". Due to this loss of correlation between adjacent time series values, the learning (prediction) task of the jammer becomes harder.

4.2 Learning Algorithm

Chaotic time series are generated by deterministic dynamical systems. Therefore, in order to predict future values, the jammer has to learn the underlying nonlinear mappings of the time series. In the particular application scenario we are concerned, the time series have to be learned solely from the past observations, without prior knowledge of the dynamical system.

As the future values of a time series depend on its previous values, we choose RNNs as the learners for the jammer. Unlike traditional feed forward neural networks, RNNs have feed back connections within the layers. It is these feedback connections that allow past experience to be taken into account when predicting the subsequent steps in time series. We use Long Short Term Memory (LSTM) networks, a variant of RNNs that is not affected by the vanishing gradient problem as the jammer's learning algorithm [5].

5 Game Formulation

This section describes the transmitter-jammer interaction, modeled as a two-player static game between the jammer J and transmitter T, repeated over discrete time k. The myopic players interact over a sequence of steps (can be finite or infinite), and at each step they solve the static game and determine the actions they would play during that time step. Although the games at each time step are independent, the learner's of the jammer evolve with time, enabling more accurate predictions as time goes on.

Since both players are equipped with SDRs, the possible actions available to both players at the network layer can be defined by choosing one of the n transmission channels in order to maximize range. For the transmitter, the probability

distributions over the n channels would depend on the output of the generators as well its adversarial distortions. Since the generator values are given at each step of the game, we focus on the mechanism used to generate adversarial distortions to formulate the game actions of T. Similarly, the jammer's probability distributions would depend on the output of the learning algorithms as well as its own adversarial distortions. As the outputs of the learning algorithms are beyond the jammer's control, we focus on its adversarial distortion mechanism to formulate its game actions.

5.1 Player Actions

Using adversarial distortions for deception of the other player comes with consequences. For the transmitter, adding adversarial distortions means using a probability distribution over the n channels that is different from what the listeners are using. This leads to unsuccessful radio transmissions (without the jammer's influence) as the listeners would be listening on different channels at a given time. For the jammer, it means using a probability distribution that is different from what the learning algorithm predicts, as a form of deception. Using a distorted probability distribution results in more unsuccessful jamming attempts as both players would be using different probabilities over the channels.

The action sets of both players are defined as different distortion severity values using an arbitrary uniform quantization:

- transmitter: $a^T = \{0.1, 0.2, \ldots, 0.9\}$
- jammer: $a^J = \{0, 0.1, 0.2, \ldots, 0.9\}$

The distortion severity value at the k^{th} time step is used to determine the adversarial distortion vectors $d^T(k)$ and $d^J(k)$ for each player obtained by sampling from a uniform distribution with different support sets. For example, the jammer would decide $a^J(k)$ and create the adversarial distortion vector $d^J(k)$ by randomly sampling from a uniform distribution over the interval $[-a^J(k), a^J(k)]$.

Since $d^T\big(a^T(k), k\big)$ and $d^J\big(a^J(k), k\big)$ are functions of the players' actions, the distorted probability distributions, $\tilde{p}^T\big(a^T(k), k\big)$ and $\tilde{p}^J\big(a^J(k), k\big)$, also become dependent on the actions of the players.

Adversarial Deviation: Adversarial deviations are scalar values ($\phi^T\big(a^T(k), k\big)$ and $\phi^J\big(a^J(k), k\big)$) that indicate how much the originally intended probability distributions $p^T(k)$ and $p^J(k)$ deviate from the actually played distributions $\tilde{p}^T\big(a^T(k), k\big)$ and $\tilde{p}^J\big(a^J(k), k\big)$. As one possible metric, we choose root-mean-square error (RMSE) to measure the adversarial deviation caused by the distortions added during the k^{th} time step:

$$\phi^T\big(a^T(k), k\big) = \Big(\frac{1}{n}\sum_{i=1}^{n}\big(p_i^T(k) - \tilde{p}_i^T\big(a^T(k), k\big)\big)^2\Big)^{\frac{1}{2}}, \tag{3}$$

$$\phi^J\big(a^J(k), k\big) = \Big(\frac{1}{n}\sum_{i=1}^{n}\big(p_i^J(k) - \tilde{p}_i^J\big(a^J(k), k\big)\big)^2\Big)^{\frac{1}{2}}. \tag{4}$$

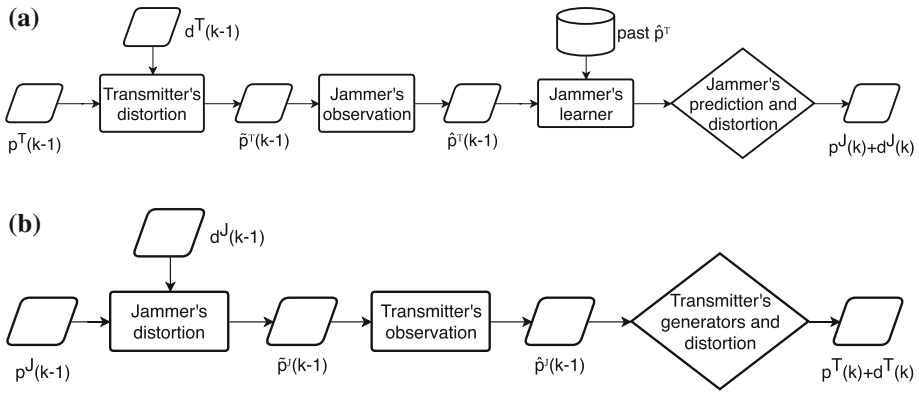


Fig. 2. (a) The decision making process of jammer based on transmitter's observed actions, (b) The decision making process of transmitter based on jammer's observed actions.

5.2 Utility Functions

At each step of the game, the players need to make two decisions (i) the probability distribution over the n channels, and (ii) the amount of adversarial distortions to introduce. As Figs. 2a and b depict, the two players use different approaches to address them. The transmitter obtains its undistorted probability distribution from the generator functions and uses the static game to decide the best adversarial distortion severity to use. The jammer observes the channel usage of the transmitter (i.e., $\hat{p}^T$) and trains n prediction models (as there is a unique generator function for each channel) to predict the next probability distribution the transmitter may use. Subsequently, similar to the transmitter, it decides on the severity of adversarial distortions based on the best response from the proposed game.

Note that the observed probabilities, $\hat{p}^T(k)$ and $\hat{p}^J(k)$, are obtained by counting the number of times each of the n channels is used by the other player during the k^{th} time interval, i.e. creating a histogram. The observed probability distributions estimate the actually played probability distributions i.e., $\tilde{p}^T(a^T(k), k)$ and $\tilde{p}^J(a^J(k), k)$ closely as the length of the time interval increases. Since the observed probabilities depend on the distorted probability distributions, we define the observation function l as $\hat{p}(k) = l(p(k), a(k), k)$, making each $\hat{p}^T(k)$ and $\hat{p}^J(k)$ dependent on $a^T(k)$ and $a^J(k)$ respectively.

Utility of the Transmitter: For the transmitter to evade jamming, the jammer's estimation of transmitter's strategy $\hat{p}^T(k)$ should be significantly different from the actual generated strategy $p^T(k)$ during the k^{th} interval. Since $\hat{p}^T(k)$ is merely the observation of $\tilde{p}^T(k)$, the transmitter can also calculate it. We use the Kullback-Leibler divergence to calculate the statistical distance between two probability distributions as:

$$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}. \tag{5}$$

The transmitter's utility function is defined as:

$$\begin{aligned} U^T(k) &= D_{KL}\big(\hat{p}^T(k)||p^T(k)\big) - \phi^T\big(a^T(k), k\big) \\ &= D_{KL}\big(l(p^T(k), a^T(k), k)||p^T(k)\big) - \phi^T\big(a^T(k), k\big). \end{aligned} \tag{6}$$

Utility of the Jammer: Large adversarial distortions from the transmitter would make the jammer's learning algorithms learn incorrect representations of the transmitter's generator functions. Therefore, by adding adversarial distortions to its predicted values, the jammer expects to deceive the transmitter into using less severe adversarial distortions. The jammer, hence, attempts to make $\tilde{p}^J(k)$ diverge from $\hat{p}^T(k)$ using its own adversarial distortions at a cost of jamming efficiency. The jammer's utility function is defined as:

$$U^J(k) = D_{KL}\big(\hat{p}^T(k)||\tilde{p}^J\big(a^T(k), k\big)\big) - \phi^J\big(a^J(k), k\big). \tag{7}$$

In the above formulations ϕ^T and ϕ^J are the adversarial deviations explained in Sect. 5.1 that penalize large adversarial distortions.

5.3 Nash Equilibrium Solution of the Game

A bi-matrix game is represented by two $(m \times n)$ matrices, $A = \{a_{i,j}\}$ and $B = \{b_{i,j}\}$ where each pair of entries $(a_{i,j}, b_{i,j})$ denotes the outcome of the game corresponding to a particular pair of decisions made by the players. These entries in the matrix are populated by the players' utility functions, U^J and U^T. A pair of strategies $(a_{i^*,j^*}, b_{i^*,j^*})$ is said to be a non-cooperative Nash equilibrium outcome of the bi-matrix game if there is no incentive for any unilateral deviation by any one of the players. While it is possible to have a scenario where there is no Nash equilibrium solution in pure strategies, there would always be a Nash equilibrium solution in mixed strategies [10].

A player is said to use a mixed strategy when it chooses to randomize over a set of finite actions. Therefore, a mixed strategy can be defined as a probability distribution that gives each of the available actions a likelihood of being selected. At each step k of the game, the two players decide on their actions by computing the Nash equilibrium solutions in mixed strategies using their corresponding utility matrices. For example, Fig. 3 shows the combined utility matrix for the two players when $k = 60$. In this particular step of the game, the best responses of both players yield a pure strategy Nash equilibrium solution, which is $a^J = 0.2$ and $a^T = 0.1$.

6 Simulation Results

In the simulations, three time series are used to generate the probabilities for each of the channels from the transmitter's perspective. We use three popular

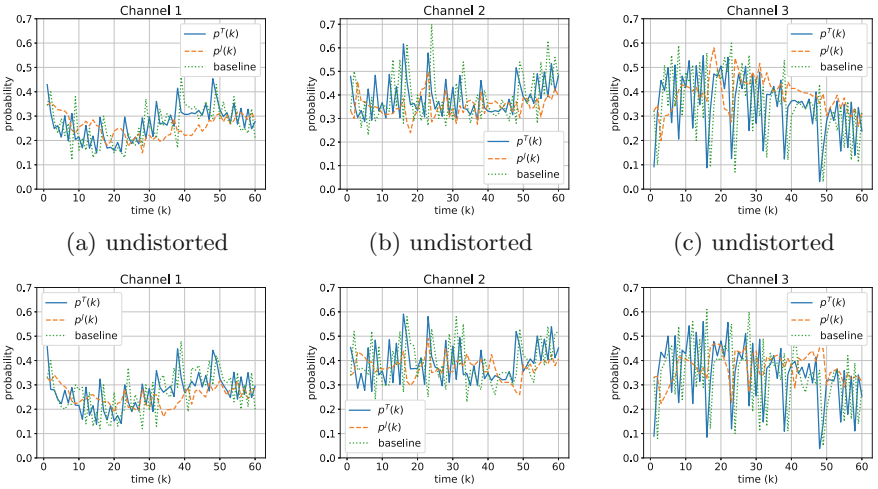
		transmitter - a^T			
		0.1	0.2	0.3	0.4
Jammer - a^J	0	(0.050,0.017)	(0.028,0.023)	(0.082,0.022)	(0.034,0.003)
	0.1	(0.038,0.005)	(0.019,0.001)	(0.073,0.016)	(0.009,0.011)
	0.2	**(0.096,0.049)**	(0.030,0.013)	(0.041,0.006)	(0.008,0.013)
	0.3	(0.048,0.083)	(0.091,0.012)	(0.062,0.000)	(0.004,0.005)
	0.4	(0.013,0.015)	(0.020,0.015)	(0.042,0.006)	(0.089,0.011)

Fig. 3. The utility matrix of the game depicting the outcomes. The jammer is the row player and the transmitter is the column player and payoffs are displayed as (jammer utility, transmitter utility).

chaotic time series, Rossler attractor, Lorenz attractor and Henon attractor as the generator functions (i.e., $g_i(k), i \in [1, 2, 3]$) for the three channels of the transmitter. At the start of the game (i.e., $k = 1$ and $k = 2$), where there are no prior observations, the jammer uses normalized probability distributions sampled from $\mathcal{U}(0, 1)$ to use over the n channels. Subsequently, after the learners commence the learning process, the predicted probability distributions $p^J(k)$ are used. During the initial stages of the game, where there are very few observations, the predictions of the online learning algorithms would not be accurate as machine learning based prediction models need sufficient data to learn the correct underlying patterns. After a certain threshold, the models are expected to stabilize and provide accurate predictions.

We train a four neuron LSTM model for each channel c_i using the observed probability distributions $\hat{p}^T$ as the training data. The performance of the LSTMs is compared against a baseline persistence algorithm which predicts the observation of the previous time step as the prediction for the current time step (i.e., $p^j(k) = \hat{p}^T(k-1)$). Figure 4 shows the probability distributions used by the transmitter, the jammer and the baseline algorithm over the three channels under two conditions. The top row shows the probability distributions when the transmitter is not introducing distortions ($a^T(k) = 0, \forall k \in \mathbb{Z}_{\geq 0}$), and the bottom row shows the distributions when $a^T(k)$ is fixed at 0.4 for all time steps. Comparing the undistorted probability distributions with the distorted probability distributions show that by introducing adversarial distortions, the transmitter can make each time series behave more abruptly, deviating from the patterns governed by the underlying equations.

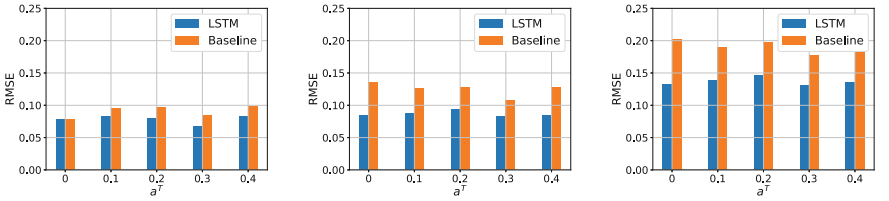
Figure 5 shows the root-mean-square error (RMSE) between the transmitter's distorted probability distribution ($\tilde{P}^T(k)$) and the jammer's predicted probability distribution ($p^J(k)$), and the baseline persistence algorithm under different adversarial distortion severities. As the LSTMs are trained online, during the early stages of the game they will not be able to make accurate predictions due to the lack of data. But as more data becomes available, the LSTMs are able to learn the underlying patterns of the data make more accurate predictions. Therefore, we show the RMSE values for the last 50% of the time series data as they reflect the true prediction capabilities of the LSTMs. We observe that



(a) undistorted (b) undistorted (c) undistorted

(d) $a^T(k) = 0.4$ (e) $a^T(k) = 0.4$ (f) $a^T(k) = 0.4$

Fig. 4. The probability distributions used by the transmitter, jammer with the deep learner and a jammer with the baseline algorithm on the three channels. The figures on the bottom row show the probability values when $a^T(k)$ is set to 0.4.



(a) Prediction error on c_1 (b) Prediction error on c_2 (c) Prediction error on c_3

Fig. 5. RMSE values of the LSTMs and the baseline algorithm for the probability distributions used on channels 1 to 3. The RMSE is calculated for the last 50% of each time series.

the performance of the LSTMs in predicting the time series are comparatively higher than the baseline persistence algorithm. The performance differences are quite significant on channels 2 and 3 where the two chaotic functions used appear more random compared to that of channel 1.

7 Conclusions and Future Work

This paper models the interaction between a sophisticated jammer and a transmitter in a search and rescue scenario as a stochastic game. The transmitter, who has a pre-shared key with the rescuers, uses stochastic channel hopping according to some pseudo-random generators as the anti-jamming mechanism.

The jammer attempts to learn the above pseudo-random generators by using online machine (deep) learning methods. In order to prevent the jammer from learning the pseudo-random generators, the transmitter introduces adversarial distortions. Similarly, to deceive the transmitter into thinking that the jammer has not been able to learn the underlying patterns, the jammer also distorts its jamming patterns on purpose. The empirical results suggest that transmitter can in fact hinder the learning capabilities of the jammer by using adversarial distortions.

Directions for future work include: comparing different (e.g. cryptographic) random number generating algorithms for the transmitters, and formulating different games, e.g. over a finite horizon or with additional solution concepts.

References

1. Boeing, G.: Visual analysis of nonlinear dynamical systems: chaos, fractals, self-similarity and the limits of prediction. Systems **4**(4), 37 (2016)
2. Cong, L., Songgeng, S.: Chaotic frequency hopping sequences. IEEE Trans. commun. **46**(11), 1433–1437 (1998)
3. Dabcevic, K., Betancourt, A., Marcenaro, L., Regazzoni, C.S.: Intelligent cognitive radio jamming - a game-theoretical approach. EURASIP J. Adv. Sig. Process. **2014**(1), 171 (2014)
4. Han, M., Xi, J., Xu, S., Yin, F.L.: Prediction of chaotic time series based on the recurrent predictor neural network. IEEE Trans. Sig. Process. **52**(12), 3409–3416 (2004)
5. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
6. Hu, H., Liu, L., Ding, N.: Pseudorandom sequence generator based on the chen chaotic system. Comput. Phys. Commun. **184**(3), 765–768 (2013)
7. Kawahara, Y., Sugiyama, M.: Change-point detection in time-series data by direct density-ratio estimation. In: Proceedings of the 2009 SIAM International Conference on Data Mining, pp. 389–400. SIAM (2009)
8. Laufer, C.: The Hobbyist's Guide to the RTL-SDR: Really Cheap Software Defined Radio. CreateSpace Independent Publishing Platform (2015)
9. Liu, S., Yamada, M., Collier, N., Sugiyama, M.: Change-point detection in time-series data by relative density-ratio estimation. Neural Netw. **43**, 72–83 (2013)
10. Nash, J.: Non-cooperative games. Ann. Math. **54**, 286–295 (1951)
11. Pathak, J., et al.: Hybrid forecasting of chaotic processes: using machine learning in conjunction with a knowledge-based model. CoRR abs/1803.04779 (2018)
12. Pual, O., Akta, I., Schnelke, C.J., Abidin, G., Wehrle, K., Gross, J.: Machine learning-based jamming detection for IEEE 802.11: Design and experimental evaluation. In: Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014, p. 110 (June 2014)
13. Xu, W., Wood, T., Trappe, W., Zhang, Y.: Channel surfing and spatial retreats: defenses against wireless denial of service. In: Proceedings of the 3rd ACM Workshop on Wireless Security, pp. 80–89. Wise 2004 (2004)
14. Zhu, Q., Saad, W., Han, Z., Poor, H.V., Başar, T.: Eavesdropping and jamming in next-generation wireless networks: a game-theoretic approach. In: Military Communications Conference, 2011-MILCOM 2011, pp. 119–124. IEEE (2011)